ISSN 0868-6157

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ

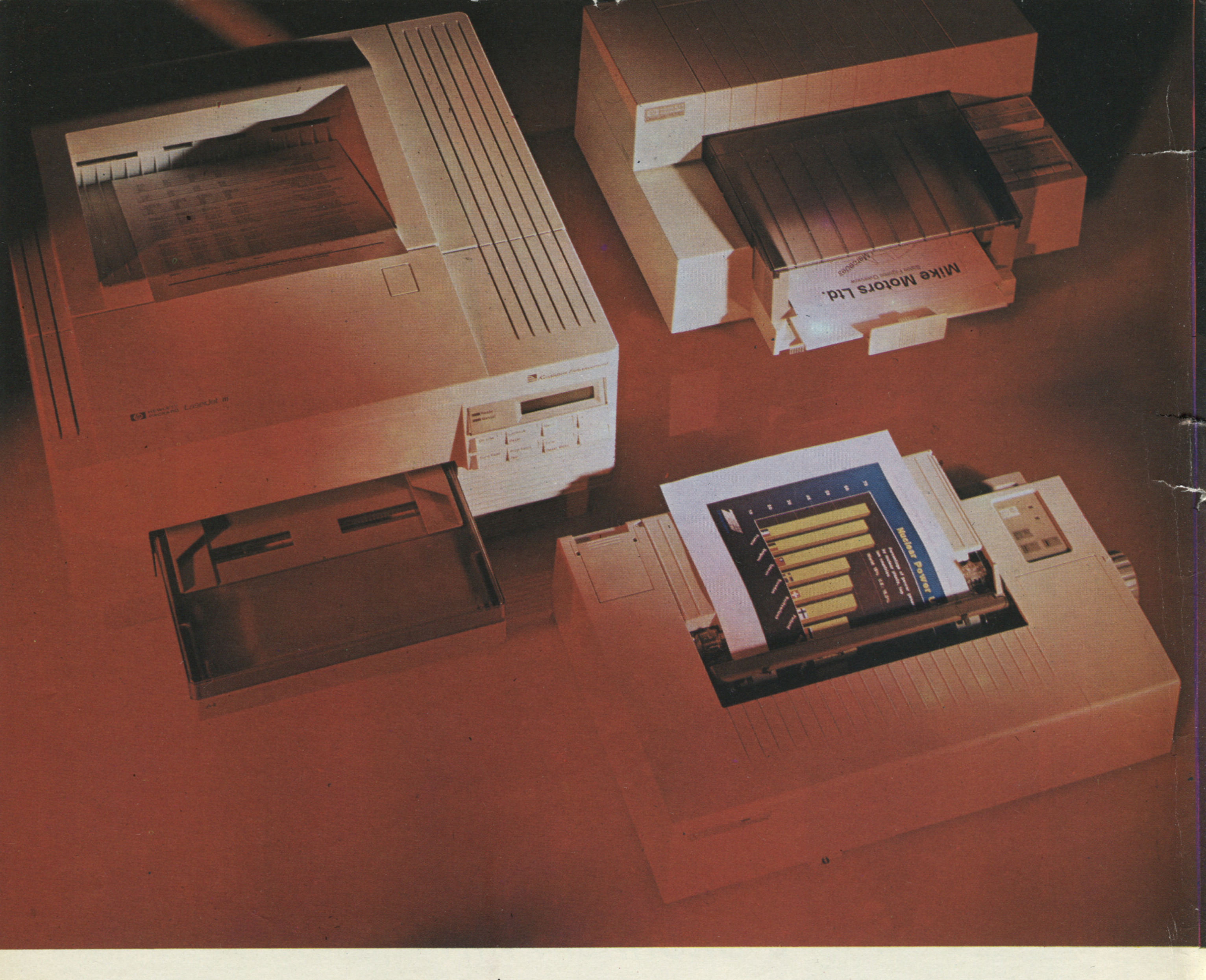
33MHz

Как создать оконный интерфейс

PC 386/33 на любой вкус

Парад СУБД

1'91



Если Вы не очень богаты — Вам не следует покупать дешевые вещи!

Технический центр фирмы Delta Group предлагает широкий выбор персональных компьютеров, периферийных устройств и программного обеспечения.

Технический центр фирмы Delta Group имеет консигнационный склад электронной оргтехники.

Технический центр фирмы Delta Group реализует оборудование фирмы Hewlett-Packard с трехлетней гарантией и последующей поддержкой.

Технический центр фирмы Delta Group проводит гибкую ценовую политику.

DELTA
DG
GROUP

Технический Центр:
Москва, ул. Осипенко, д. 15, кор. 2, офф. 207.
Телефон: 230-56-12 Факс: 230-21-82

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ



1'91

Совместное советско-американское предприятие «СОВАМИНКО»
КОМПЬЮТЕР ПРЕСС
ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ

Адрес редакции:
113093, г.Москва, аб.ящик 37
Тел. для справок: 150-17-03
Бюро рекламы: 156-81-33
Факс: 200-22-89

ДОРОГОЙ ЧИТАТЕЛЬ!

Редакция журнала чрезвычайно благодарна тебе за то, что несмотря на все трудности, несмотря на нехватку товаров первой необходимости, несмотря на постоянную неуверенность в завтрашнем дне, ты находишь время и силы на чтение научно-популярной литературы. Нам особенно приятно сознавать, что интерес к событиям, происходящим в мире персональных компьютеров, ты стремишься удовлетворить с помощью журнала «КомпьютерПресс». Ведь ни для кого не секрет, что благополучие, да и само существование любого печатного органа зависит, в конечном счете, только от тебя — от твоих вкусов и пристрастий. Мы ждем предложений и пожеланий от наших старых читателей и приветствуем всех, кто решит подписаться на наш журнал в наступившем году. Сделать это не поздно. Условия годовой подписки, а также бланк заказа на получение журнала «КомпьютерПресс» наложенным платежом ты найдешь на страницах 79 и 80.

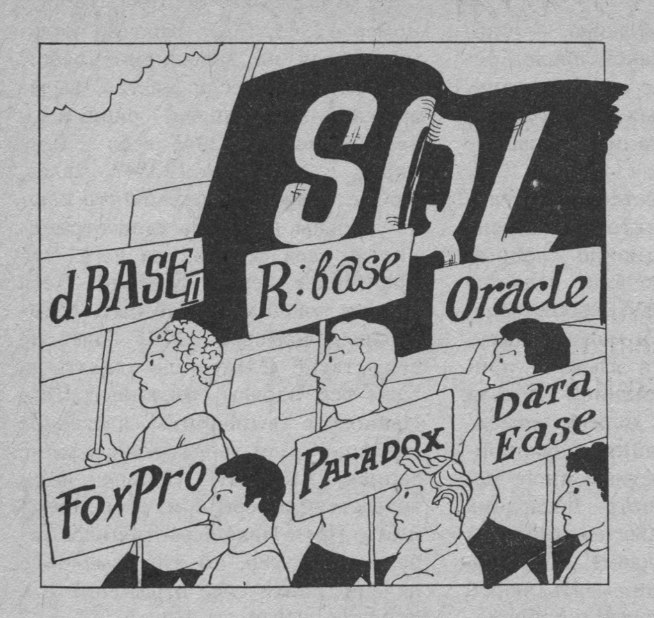
Редакция журнала желает тебе ни при каких обстоятельствах не терять присутствия духа и оставаться любознательным, НЕСМОТРЯ НИ НА ЧТО!

Годовая подписка на наш журнал —

это экономия вашего времени!

Сдано в набор 3.01.91. Подписано к печати 14.01.91. Формат 84x108/16. Печать офсетная.
Усл.печ.л.8,4+0,32 (обл.). Тираж 100 000 экз. (1 завод–55 000). Заказ 2125. Цена 3 р. 15 к.

Типография издательства «Калининградская правда»
236000, г.Калининград, ул.Карла Маркса, 18



Открывающиеся перспективы были ослепительны.

А. и Б. Стругацкие
"Понедельник начинается в субботу"

Парад СУБД

Можно сказать, что до середины восьмидесятых годов системы управления базами данных находились в том состоянии, которое по образному выражению создателя теории научно-технических революций, Томаса Куна, называется предпарадигматической фазой. Поставщики СУБД предлагали огромный набор различных решений. Практически каждая СУБД имела собственные, отличные от других, назначение и архитектуру, и каждая имела свои достоинства и недостатки. Но ни одна из них не могла претендовать на роль стандарта. Попытки создания таких стандартов для больших универсальных ЭВМ, среди которых наиболее известна СУБД CODASYL, не имели серьезного успеха.

В области микрокомпьютеров разработчики пытались создать базы данных на все случаи жизни, которые были бы одновременно и столь просты как PFS:FILE, и обладали столь богатыми возможностями как dBASE. Поставщики произносили заклинания типа "реляционная база данных" и "SQL", на практике весьма вольно подходя к трактовке этих понятий. Объяснить их шаманство довольно просто — без подобных определений ни одна из СУБД не имела бы успеха на рынке.

МОДЕЛИ: ЗАМЕНЯТ ЛИ ОБЪЕКТЫ ОТНОШЕНИЯ?

Реляционная модель

Последнее десятилетие ознаменовало собой триумф модели реляционной, которая из красивой научной абстракции, придуманной задолго до этого Э.Ф. Коддом, превратилась в средство реализации баз данных. И как каждая идея, долго боровшаяся за первенство, реляционная модель превратилась в абсолют. Очевидно, на протяжении ближайших пяти-десяти лет в этой области ничего радикально не изменится — модель постараются усовершенствовать, будут создаваться новые более производительные алгоритмы, новые дополнения, позволяющие использовать модель в решении нетрадиционных для нее задач и т.п.

В апреле этого года Кодд опубликовал книгу, в которой он описал "вторую версию" старой реляционной модели. Он расширил до 300 список требований (ранее их было только 15), которым должна удовлетворять модель, чтобы называться реляционной. Дополнительные требования не изменяют существа реляционной модели, а касаются представления данных и языка запросов. Они отражают то новое, что мы узнали о реляционной модели со дня ее опубликования.

Говоря о будущем реляционной модели, следует отметить, что ей остается избавиться от одного уязвимого места — использования в технике, — и тогда ей суждены долгие лета.

Чем же так хороша реляционная модель? Чтобы ответить на этот вопрос, обратимся к теории реляционных баз данных, после чего перейдем к практике их реализации.

Базы данных родились в недрах больших универсальных ЭВМ. На заре своего существования компьютеры были чрезвычайно дороги, вычислительные мощности и

внешние запоминающие устройства составляли довольно внушительную часть затрат предприятий. И как только вставал вопрос о реализации сложной обработки данных, первым делом стремились использовать машинное время и память с максимальной эффективностью. И именно поэтому получил широкое распространение иерархический подход к организации баз данных. Иерархические базы данных имеют форму деревьев с дугами-связями и узлами-элементами данных. Иерархическая структура предполагала неравноправие между данными — одни были жестко подчинены другим. Подобные структуры, безусловно, четко удовлетворяют требованиям многих, но далеко не всех реальных задач. Для того, чтобы реализовать в базе наряду с вертикальными и горизонтальные связи, разработали сетевую модель данных, которая, правда, унаследовала многие недостатки иерархической и главный из них, необходимость четко определять на физическом уровне связи данных и столь же четко следовать этой структуре связей при запросах к базе. В современных системах, где человек все более и более отгораживается от аппаратных средств компьютера многочисленными дружелюбными интерфейсами, такой подход, безусловно, неприемлем.

Реляционная модель появилась вследствие стремления сделать базу данных как можно более гибкой, и действительно, эта модель предоставила простой и эффективный механизм поддержания связей данных.

Во-первых, все данные в модели представляются в виде таблиц, и только таблиц. Сравните реляционную модель с иерархическими и сетевыми моделями или более современными типа "сущность-связь", и вы убедитесь, что слово "только" также важно в определении, как и слово "таблица". Реляционная модель — единственная из всех, обеспечивающая единообразие представления данных. И сущности, и связи этих самых сущностей представляются в модели совершенно одинаково, — таблицами. Правда, такой подход усложняет понимание смысла хранящейся в базе данных информации, и, как следствие, манипулирование этой информацией.

Избежать трудностей манипулирования позволяет второй элемент модели — реляционно полный язык (отметим, что язык является неотъемлемой частью любой модели данных, без него модели не существует). Полнота языка в приложении к реляционной модели означает, что он должен выполнять любую операцию реляционной алгебры или реляционного исчисления (а полнота последних вполне доказана (математически) все тем же Э.Ф.Коддом). Более того, язык должен описывать любой запрос в виде операций с таблицами, а не с их строками. Одним из таких языков является SQL.

Сами по себе первые две составляющих реляционной модели не слишком ограничивают разработчиков. Практически любая файловая система может быть в этом случае реляционной базой данных, если ее язык позволяет обращаться к файлам, а не искать отдельные записи. Такое отсутствие точности и позволило многим фирмам заявлять, что их система самая реляционная из всех реляционных.

Однако существует и третий элемент реляционной модели, который практически не учитывается ни в одной из существующих систем. Этот элемент требует от реляционной модели поддержания некоторых ограничений целостности. Одно из таких ограничений утверждает, что каждая строка в таблице должна иметь некий уникальный идентификатор, называемый первичным ключом. Второе ограничение накладывается на целостность ссылок между таблицами. Оно утверждает, что атрибуты таблицы, ссылающиеся на первичные ключи других таблиц, должны иметь одно из значений этих первичных ключей. Тут то и лежит камень преткновения создававшихся до недавнего времени систем. Представьте себе, что некий рабочий Петя Иванов имеет табельный номер 12345. Этот рабочий получил наряд на выполнение работ Что случится, если мастер, выдававший этот наряд, ошибся и вместо номера 12345 ввел в базу наряд с номером 12354? Далее, любые изменения первичного ключа должны находить свое отражение во всех ссылках на него. Что произойдет, если рабочий Петя Иванов уволился? Рабочего с табельным номером 12345 более не существует. Следует ли удалять из базы все наряды, выданные Пете Иванову, (выполнить каскадное удаление) или запретить удаление записи РАБОЧИЙ? Наконец, если вследствие очередной реорганизации у Пети Иванова изменился табельный номер, то это изменение должно найти свое отражение и в нарядах, выданных Пете.

Вопреки распространенному убеждению ограничения целостности поддерживались СУБД больших машин еще до появления реализаций реляционных баз данных. В то же время, созданные значительно позже реляционные СУБД, и, особенно, СУБД для микрокомпьютеров, такую способность утратили. Теперь для поддержания ограничений целостности пользователю приходится писать прикладные программы.

Сегодня известны два основных пути, по которым можно достичь поддержки ограничений целостности: первый — это так называемые "триггеры", которые являются составной частью базы данных и выполняются автоматически при изменении соответствующих элементов данных. Разные модификации этого подхода реализованы в сервере SQL, а также СУБД Ingres. Преимущество метода состоит в том, что он позволяет реализовывать довольно сложные правила. Альтернативный подход — создание декларативного языка описания данных, позволяющего специфицировать и ограничения целостности. Этот метод реализован в СУБД Extended Edition Data Manager, SQL/DS и DB2 фирмы IBM. Преимущество декларативного подхода состоит в том, что с его помощью пользователю легче зада-

вать ограничения целостности. Комитет ANSI по SQL использует в своей версии декларативный подход.

Стандарт SQL

Точно также как реляционная модель покорила мир баз данных, SQL покорил мир языков реляционных баз данных. Очевидно, язык QUEL, использовавшийся ранее в Ingres, по ряду параметров превосходил SQL, но к настоящему времени он практически исчез из обихода. И в Ingres теперь используется SQL, и даже Ashton-Tate попыталась использовать этот язык.

Версия SQL, разработанная IBM, де-факто стала еще одним стандартом. Для подавляющего большинства независимых производителей программного обеспечения значительно важнее обеспечивать совместимость с IBM, чем со стандартом Американского национального института стандартов (ANSI). В то же время, с точки зрения поддержания ограничений целостности, IBM опережает стандарт. Комитет ANSI разработал стандартный SQL в 1986 году и переработал его в 1989. Несмотря на то, что все реляционные СУБД сегодня поддерживают SQL, все эти SQL несколько отличаются друг от друга, и ни один из них не соответствует стандарту ANSI89. Для проверки на соответствие стандарту Национальный институт стандартов и технологии США (ранее эта организация называлась Национальным бюро по стандартам) разработал специальный тест. Фирма Oracle заявила, что седьмая версия одноименной СУБД прошла этот тест.

Вместе с тем, все поставщики отмечают, что соответствие стандарту еще не означает совершенства, поскольку в стандарте не учтены все те же вопросы поддержания ограничений целостности и, например, не реализована операция внешнего соединения. И то, и другое требуется практически в каждой прикладной задаче.

Во второй версии стандарта SQL2 эти дополнения, очевидно, будут учтены. Но, к сожалению, в него внесено так много других изменений, что вряд ли какая-нибудь из фирм сможет реализовать этот стандарт.

По мнению авторитетных экспертов, к числу которых относятся Майк Стоунбрэйкер (Mike Stonebreaker), вице-президент компании Ingres и Крис Дейт (Chris Date), известный нам по книге “Введение в системы баз данных“, стандарт SQL2 нельзя назвать удачным.

Объектная ориентация

Этот вопрос мы уже рассматривали (см. КомпьютерПресс № 11, 1990), поэтому остановимся на нем лишь вкратце. Новые области использования вычислительной техники, как то научные исследования, автоматизированное проектирование и автоматизация учреждений, потребовали от баз данных способности хранить и обрабатывать новые объекты — текст, аудио- и видеоинформацию, а также документы. Все это не так просто, поэтому стали поговаривать о создании преемника реляционной — объектно-ориентированной модели данных. Никто не может дать ей точного определения. Дейт и Стоунбрейкер считают, что все преимущества объектной ориентации можно извлечь из существующей реляционной модели, тогда как Якоб Стейн (Jacob Stein) из Servio Logic утверждает обратное: реляционная модель не может служить основой для создания объектно-ориентированной хотя бы потому, что манипулировать объектами на основе теории отношений невозможно. Отметим, что Servio Logic уже сейчас поставляет на рынок объектно-ориентированную СУБД Gemstone. В целом же, несмотря на множество околонаучных разговоров по поводу объектно-ориентированных систем и их многочисленные преимущества, — реализация сложных типов данных, связь с языками программирования и т.п., — на ближайшее десятилетие господство реляционных СУБД гарантировано.

АРХИТЕКТУРА: РАСПРЕДЕЛЯЙ И ВЛАСТВУЙ...

Архитектура типа “клиент-сервер“

В восьмидесятых годах получили признание архитектуры типа “клиент-сервер“. Основной тенденцией при создании фронтальных средств является то, что они все более ориентируются на персональные компьютеры с графическими интерфейсами, обеспечивающие пользователю максимальное удобство работы. Второй аспект этого движения состоит в создании средств коммуникации с серверами баз данных, и прежде всего с сервером DB2, реализованном на больших универсальных компьютерах. Кроме того, разрабатываются интерфейсы для таких специализированных машин баз данных, как Sybase и Gupta.

Основной трудностью в создании коммуникационных языков для систем “клиент-сервер“ остается все то же несовершенство стандарта SQL, о чем уже было сказано.

Компании прорываются сквозь эту трудность как поодиночке, так и сообща. Они создали специальную группу SQL Access Group, в которую входят пятнадцать компаний, включая Ingres, Informix и Oracle. Вместе с тем, ряд компаний создают собственные нестандартные интерфейсы: Oracle — SQLConnect, Sybase — процедурный интерфейс для взаимодействия с удаленными базами данных, Ingres — General Communication Architecture, Apple — CL/1. Наиболее успешно трудности коммуникации обходит фирма Information Builders. Ее СУБД Focus имеет шлюзы к огромному количеству реляционных и нереляционных СУБД, в рамках единой архитектуры как для главного компьютера, так и для системы “клиент-сервер“, также поставляется средство для подключения сервера SQL.

Распределенные базы данных

Следующий логический шаг от систем "клиент-сервер", безусловно, приведет к распределенным базам данных. Можно сказать, что база данных будет только тогда распределенной, когда пользователю не нужно будет знать, что она распределена по нескольким узлам, это свойство называется прозрачностью.

Ни одна из фирм сейчас не поставляет распределенных баз данных, хотя многие к этому стремятся. Например, Oracle, Informix и Ingres, позволяют выполнять операцию соединения данных, расположенных в разных узлах сети, а две последние даже пытаются оптимизировать затраты на обработку запроса. Sybase, Interbase и Ingres поддерживают двухфазный протокол блокировки, позволяющий синхронизировать доступ различных транзакций к одним и тем же данным. Конечно, не все пользователи знают, для чего нужны распределенные базы данных, а кое-кто из разработчиков даже считает, что они вообще не нужны. Но и те, и другие активно стремятся к тому, чтобы такие базы данных все же появились. Вообще говоря, это направление содержит еще много загадок, и не исключено, что окажется выгоднее хранить единственную копию базы данных на одном большом компьютере, а всем остальным пользователям сети просто раздавать ответы на их запросы. Ну что ж, как говорят англичане, — live and see, (поживем — увидим).

ФИРМЫ: РУКОВОДЯЩЕЙ РОЛИ IBM ПОКА НИКТО НЕ ОТМЕНЯЛ...

Сдвиг в сторону создания распределенных баз данных изменил позиции лидеров, предоставив возможности занять позиции на рынке другим компаниям, ранее специализировавшимся на создании СУБД для универсальных компьютеров. Изменения продолжаются, поэтому сейчас довольно сложно говорить об окончательной расстановке сил.

Компания Ashton-Tate

Ashton-Tate занимает особое место на рынке СУБД для персональных компьютеров. Компания раньше всех вступила в эту сферу. Выпустив в самом начале эры микрокомпьютеров свой пакет dBASE и оставаясь в течение долгого времени непревзойденной, она привязала к себе большинство пользователей микрокомпьютеров. По многим позициям Ashton-Tate и сейчас не теряет лидерства, однако, в последнее время чувствуется, что компания запаздывает с внедрением изменений и, очевидно, понесет потери в связи с распространением новых аппаратных средств и операционных систем. Отметим, например, что к моменту появления сервера SQL для OS/2, Ashton-Tate не смогла реализовать стандартного SQL в dBASE IV, пообещав это сделать только в версии 1.1., вследствие чего фирма IBM отобрала у компании права дистрибутора сервера. Очевидно, Ashton-Tate находится на пороге серьезной реорганизации. Однако, все же компания реализует целый ряд программных продуктов, предназначенных для работы с большими системами, и обещает создать в ближайшем будущем значительно более мощную версию dBASE. Учитывая лояльность пользователей и огромное количество проданных экземпляров dBASE (сегодня число установок всех версий dBASE достигло фантастической цифры — около 6 миллионов), компания, очевидно, сохранит свои лидирующие позиции.

Компания Oracle

Пожалуй, Oracle символизирует собой самый крупный успех в разработке программного обеспечения за все восьмидесятые годы. Предположительно, объем продаж этой компании превысил в 1990 финансовом году 1 миллиард долларов. Компания образовалась в 1979 году и выпускала один единственный продукт — ORACLE, СУБД с языком SQL. Феерический успех Oracle объясняется, очевидно, тем, что, с одной стороны, эта компания реализовала идею IBM раньше самой IBM, а с другой стороны, ORACLE реализовывалась для миникомпьютеров и рабочих станций, т.е. в тех аппаратных средах, где IBM традиционно не имела лидерства, и лишь впоследствии были выполнены разработки для больших ЭВМ и персональных компьютеров.

Получая от продажи программ для персональных компьютеров всего лишь 5—10 процентов своей прибыли, Oracle вряд ли может претендовать на роль лидера в рассматриваемой нами области. Однако учитывая то, что СУБД Oracle функционирует практически на всех типах компьютеров и обеспечивает полную переносимость исходных текстов, компания имеет отличные перспективы на рынке распределенных СУБД. Существуют аппаратные средства, позволяющие переносить тексты из одной машины в другую, тем самым избавляя пользователя от необходимости приобретать или писать заново программы. Oracle поставляет к своим базам данных для PC сервер, а программы и данные могут переноситься с компьютера на компьютер без всяких изменений. Начав с больших компьютеров, и постепенно двигаясь в сторону PC, Oracle достигла двух важных целей: она стала поставлять модель для распределенной обработки данных и приобрела популярность среди пользователей универсальных ЭВМ. Эти пользователи имеют свои специфические

потребности, связанные с защитой данных и распределением информации между центральными и удаленными рабочими местами. Поставщики СУБД для PC двигаются как раз в противоположном направлении туда, где Oracle уже крепко заняла свои позиции и гарантировала своим пользователям, что дальнейшего усложнения системы команд не произойдет, чего не могут обещать другие, т.к. идти от большего к меньшему значительно легче, чем наоборот. Кто в этом движении окажется первым, покажет время. Но и сейчас можно с уверенностью сказать, что Oracle с ее богатыми традициями в области универсальных ЭВМ и вычислительных сетей, безусловно, расширит свой плацдарм на рынке микрокомпьютеров.

Итак, если вы хотите работать с гетерогенной распределенной СУБД, то не забудьте про Oracle, она заслуживает серьезного отношения.

Компания Microrim

Знаменитая СУБД R:base фирмы Microrim имеет репутацию хорошо спроектированного микрокомпьютерного продукта с весьма обширными возможностями. R:base пока работает только с компьютерами класса PC, однако Microrim стремится к тому, чтобы создать версию R:base, работающую в среде версии Presentation Manager для OS/2. Версии для VAX/VMS и UNIX, а также для компьютеров Apple сейчас находятся в процессе разработки. Microrim также разрабатывает средство под названием Vanguard, предназначенное для гетерогенной обработки данных. В среде Vanguard пользователи смогут распределять и коллективно использовать данные, расположенные на различных машинах, операционных системах и СУБД, обращаясь к ним из различных точек сети. Возникновение такого проекта говорит о том, что Microrim более чувствительна к потребностям изменяющегося рынка по сравнению с конкурентами.



Компания Information Builders

Разработчик СУБД FOCUS для PC, прошел путь во многом аналогичный Oracle, — от универсальных машин к микрокомпьютерам, — и теперь реализовал свою систему в среде большинства моделей компьютеров. FOCUS для PC, хотя и довольно сложен в освоении, обладает всеми преимуществами мощной базы данных. Он поставляется для MS-DOS и OS/2, а также для большинства больших вычислительных машин, обеспечивая средства распределенной обработки данных. FOCUS был одним из первых продуктов, который устанавливал связь между мини- и большим компьютером в собственной среде. Многопользовательская версия PC/FOCUS поддерживает средства параллельного доступа в локальных сетях. Наконец, учитывая, что к настоящему времени продано более 500.000 экземпляров FOCUS для больших компьютеров, Information Builders вполне можно рассматривать в качестве лидера рынка СУБД для больших ЭВМ и серьезного конкурента на рынке микрокомпьютеров.

Компания Borland

Простота использования средств расширения Paradox повысила мощь этой системы. Теперь сочетание простоты и мощности сделало Paradox значительно более популярной системой, чем это было, скажем, в 1985 году, когда Paradox впервые появился на рынке как продукт фирмы Ansa Software. После того, как Borland купила Ansa, она продолжала совершенствовать Paradox. В недалеком будущем предполагается появление версии с SQL.

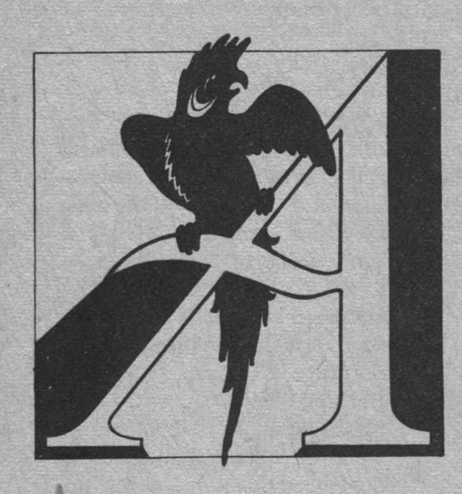
Компания IBM

Голубой гигант отнюдь не борется за пальму первенства в области производства СУБД для микрокомпьютеров. Ему слишком тесно в этой малой нише рынка. Но как и везде, где речь идет о вычислительной технике, без IBM трудно обойтись. Дело в том, что для пользователей как персональных компьютеров, так и больших машин одним из наиболее важных стандартов является язык манипулирования базой данных SQL. Несмотря на то, что на протяжении последних 20 лет было реализовано достаточно много языков управления реляционными базами данных, фактическим стандартом все же стал SQL фирмы IBM. Он стал стандартом вопреки весомой критике со стороны целого ряда признанных экспертов по базам данных. SQL, как можно догадаться, хорошо привился просто в силу политических обстоятельств: любой другой язык СУБД попросту не поддержала бы IBM, что равносильно невозможности использования его в качестве стандарта.

IBM разработала на основе своего SQL несколько СУБД. К их числу относятся наиболее известная DB2, предназначенная для больших машин с операционной системой MVS, и SQL/DS для малых универсальных

компьютеров, работающих под операционными системами VM или VSE, ведется разработка подобных пакетов и для операционной системы UNIX. Что касается персональных компьютеров, то для расширенной версии OS/2 (Extended Edition) разработана система Data Manager, обеспечивающая полную совместимость с DB2. SQL в ближайшем будущем станет стандартом и микрокомпьютерных СУБД вследствие развития локальных сетей и систем "клиент-сервер". Так что, руководящей роли IBM пока никто не отменял...

ГЛОССАРИЙ: ТЕРМИНЫ ПО СИСТЕМАМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ



Атрибут (attribute) — минимальный элемент данных, определяющий некоторое свойство объекта данных.

Блокировка (locking) — процесс временной приостановки выполнения запросов к данным в распределенных системах и системах коллективного пользования. Блокировка выполняется для того, чтобы запросы не мешали друг другу в процессе обработки общих данных.

Внешнее соединение (outer join) — операция соединения, когда при слиянии двух отношений по равенству значения атрибута строки (кортежи) первого отношения, не имеющие равного значения атрибута во втором отношении, сохраняются в результирующей таблице.

Возрастание (ascending) — направление упорядочивания данных в базе от наименьшего к наибольшему значению в алфавитно-цифровом порядке в соответствии с кодами (ASCII, EBCDIC и т.п.) символов, принятыми в системе.

Декларативный язык (declarative language) — язык программирования или манипулирования базой данных. Отличительной особенностью декларативных языков является то, что они описывают не последовательность действий, а результат, который должен быть получен после выполнения программы.

Домен (domain) — область определения атрибута реляционной таблицы, может быть как непрерывным (например, действительные числа), так и дискретным (месяцы года или логические переменные типа истина/ложь).

Естественный язык (natural language) — язык, команды которого имеют структуру предложений одного из естественных языков (английский, русский и т.п.).

Запрос (query) — предложение, описывающее критерий поиска информации в базе данных.

Импорт (import) — считывание файла, созданного в другой программе, для использования базой данных. Большинство СУБД способно считывать форматы ASCII, DIF и WKS.

Индекс (index) — группа указателей, устанавливающих связь между значениями полей и их адресами. Большинство баз данных поддерживает множественное индексирование базы данных.

Кардинальное число (cardinality) — число кортежей отношения.

Клиент-сервер (client-server) — описывает частный случай распределенной базы данных, где под клиентом понимается фронтальная программа, обеспечивающая взаимодействие с пользователем, а под сервером — тыловая программа, обеспечивающая централизованную обработку данных.

Кортеж (tuple) — строка реляционной таблицы.

Модификация (update) — предложение, описывающее информацию в базе данных, подлежащую изменению.

Мощность — см. Кардинальное число.

Отношение (relation) — множество, каждый элемент которого включает значения n атрибутов (1 атрибут — унарное, 2 атрибута — бинарное, n атрибутов — n-арное отношение).

Отчет (report) — документ, представляемый на бумаге или экране, который содержит некоторый набор информации из базы данных.

Представление пользователя (view) — подмножество базы данных, включающее часть или все отношения исходной базы данных, а также формы отчетов и экранные формы. На отношения в представлении могут накладываться фильтры, которые обеспечивают, например, формирование временной таблицы, являющейся соединением двух таблиц базы данных.

Проекция (projection) — одноместная операция реляционной алгебры, в результате которой формируется новое отношение. Кортежи результирующего отношения представляют собой выборку из кортежей исходного отношения значений некоторых атрибутов исходного отношения.

Процедурный язык (procedural language) — язык программирования или манипулирования базой данных. Отличительной особенностью процедурных языков является то, что они описывают последовательность действий, а не результат, который должен быть получен после выполнения программы.

Реляционная алгебра (relational algebra) — средство процедурного описания запросов к реляционной базе данных. Основными операциями реляционной алгебры являются селекция, соединение и проекция.

Реляционное исчисление (relational calculus) — средство декларативного описания запросов к реляционной алгебре. Запросы описываются с помощью кортежных переменных, одноместных и двухместных предикатов и правильнс построенных формул.

Селекция (selection) — одноместная операция реляционной алгебры, в результате которой формируется отношение, кортежи которого будут включать кортежи (или только некоторые атрибуты кортежей) исходного отношения, удовлетворяющие определенному логическому условию.

Соединение (join) — двухместная операция реляционной алгебры, в результате которой формируется отношение, кортежи которого включают пары кортежей исходных отношений, атрибуты которых удовлетворяют логическому условию.

Степень отношения (degree) — число атрибутов отношения.

Таблица (table) — см. Отношение.

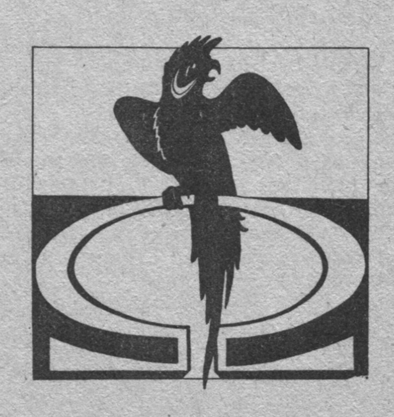
Убывание (descending) — направление упорядочивания данных в базе от наибольшего к наименьшему значению в алфавитно—цифровом порядке в соответствии с кодами (ASCII, EBCDIC и т.п.) символов, принятыми в системе.

Управление параллельным доступом (concurrency control) — метод управления запросами, коллективно использующими одни и те же данные. Одним из методов управления является блокировка.

Экспорт (export) — запись информации на диск в виде файла, предназначенного для использования в какой-либо другой программе.

Язык QBL (query by example) — язык поиска в базе данных, когда критерий специфицируется в экранной форме, описывающей поля записи.

Язык SQL (structured query language) — язык манипулирования данными, предназначенный для выполнения реляционных запросов и имеющий минимальное количество операций. Он был использован IBM в реляционной СУБД DB2. SQL практически стал стандартом, реализуемым сейчас как в больших ЭВМ, так и в микрокомпьютерах.



2 Зак. 2125

ПАРАД СУБД ПРОДОЛЖАЕТСЯ...

Фирма Aclus
Программа 4th Dimension 2.011

Системные требования. Программа 4th Dimension 2.011 функционирует на компьютерах класса Apple Macintosh под управлением операционной системы Mac System 6.0, совместима с большинством локальных сетей, включая Ethernet, 3Com, AppleTalk, Apple LocalTalk и AppleShare. Для функционирования системы требуется 1 Мбайт оперативной памяти и жесткий диск.

Характеристика программы. Программа представляет собой реляционную СУБД, обеспечивающую взаимодействие с пользователем посредством системы меню. 4th Dimension поддерживает многопользовательский режим работы и имеет интерфейс с сервером базы данных HFS-AFS. Язык СУБД имеет развитые средства программирования, включая условные операторы, циклы, передачу параметров между подпрограммами. Имеются средства отладки, встроенные генератор меню и редактор текстов, допустим оконный режим работы.

Генератор отчетов программы позволяет создавать полноэкранные отчеты путем создания бланков в графическом режиме, созданные таким образом форматы сохраняются в виде отдельных файлов. Имеется возможность использования упрощенных стандартных отчетов. При генерации отчетов можно определять поля в виде вычисляемых формул, подобно тому, как это делается в электронных таблицах.

Структура данных. 4th Dimension допускает одновременную обработку 99 файлов. Каждый файл может иметь до 16 миллионов записей по 511 символов каждая. Программа поддерживает десять типов данных, включая символьные, десятичные, целые, денежные, логические данные, а также время, дату, и шаблонное представление данных. Программа позволяет также включать в базу данных тексты объемом до 32 Кбайт. Цифровые данные могут достигать значения 2.000.000.000.

4th Dimension обеспечивает сортировку и индексирование файлов. Файл одновременно может быть отсортирован по 30 полям. Число индексов на один файл не ограничено. При описании индексов допустимо использование математических операций.

Имеются возможности экспорта и импорта файлов: допускается считывание и запись файлов в форматах DIF, SYLK, а также записей фиксированной длины в коде ASCII.

Защита данных. Допускается защита данных на уровне баз данных, файлов и отдельных полей. Кроме того, допускается определить данные, недоступные каждой из используемых прикладных программ. При регистрации в системе пользователь должен указать свой пароль.

Цена: 795 долл. США.

Фирма Alms & Plus.
Программа Information Management (1M) Level 4

Системные требования. Программа Information Management (1M) Level 4 функционирует на микрокомпьютерах класса IBM PC/AT/XT, IBM PS/2 и совместимых с ними. Допускается использование операционных систем MS-DOS и PC-DOS версии 2.1. и выше. Программа требует не менее 384 Кбайт оперативной памяти и дисковую память объемом не менее 3 Мбайт.

Программа функционирует в среде локальной сети Novell.

Характеристика программы. Программа Information Management (1M) Level 4 написана на языке Basic и является полностью меню-ориентированным средством разработки прикладных программ и баз данных. Пользователь может создавать, файлы, отчеты, программы и приложения, объединяющие в себе перечисленные выше элементы.

СУБД поддерживает реляционную модель данных, но не совместима ни с одним из серверов базы данных. К сожалению, фирма не сообщила, может ли быть использован в СУБД язык SQL.

Information Management (1M) Level 4 имеет встроенную подсистему "экранный художник" и генератор отчетов, который позволяет создавать полноэкранные формы отчетов, а также использовать систему стандартных отчетов.

Система запросов организована по методу "querybyform", когда пользователь видит на экране бланк, и на этом бланке он должен задать значения (или выражения, описывающие значения) полей, включаемых в критерий поиска.

Information Management (1M) Level 4 имеет интерфейсы с языками C, Pascal, Basic и Fortran.

Структура данных. Программа может обеспечивать одновременную обработку 6 файлов данных. Каждый файл может включать до 1 миллиона записей с максимальным числом полей на одну запись — 250, при этом максимальная длина записи не должна превышать 4.000 символов. Программа позволяет поддерживать 6 типов данных, включая текстовые/символьные и последовательные типы данных, десятичные и целые числа, а также форматы денежных данных и даты. Максимальная длина текстового поля — 76 символов, а числовые данные могут иметь до 14 знаков.

Information Management (1M) Level 4 обеспечивает сортировку и индексирование данных. Программа может выполнять сортировку файла по 5 полям. Максимальная длина индекса, который может включать одновременно не более 5 полей, составляет 50 символов. На каждый файл можно создавать не более 10 индексов. При описании индексов допустимо использование математических операций. Допускается использование структур данных в виде Вдеревьев.

Имеются возможности экспорта и импорта файлов: файлы считываются и записываются в форматах DIF, SYLK, а также в форме записей фиксированной длины в коде ASCII и записей с разделением значений запятыми.

Защита данных. Данные защищаются на уровне баз данных и файлов. Допускается определить данные, недоступные приложениям. При регистрации в системе пользователь должен указать свой пароль. Шифрование данных недопустимо.

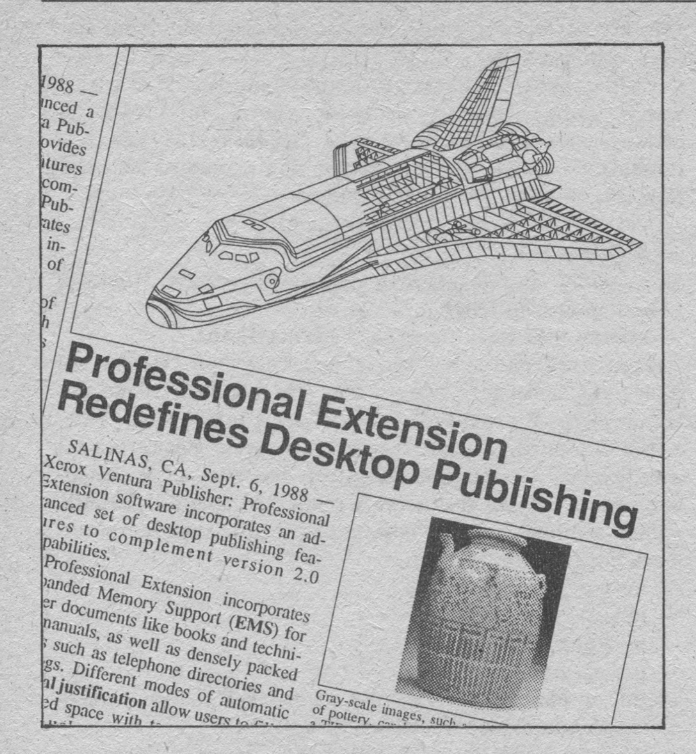
Дополнительная информация. Предоставляется гарантия сроком на 90 дней с момента продажи. Фирма заключает договоры на сопровождение пакета. Цена пакета — 695 долл. В сетевой реализации для 4 пользователей — 1.250 долл., плюс по 595 долл. на каждого дополнительного пользователя.

М.Михайлов

По материалам:

H.Edelstein, "An update on relational Technology", DataBased Advisor, June, 1990.
Datapro Reports on Microcomputers. Data Management.
G.Schussel, "The IBM Effect", DataBased Advisor, March, 1990.

(Продолжение следует)



1988 —
nced a
a Pub-
ovides
atures
com-
Pub-
ates
in-
of

Professional Extension Redefines Desktop Publishing

SALINAS, CA, Sept. 6, 1988 —
Xerox Ventura Publisher: Professional
Extension software incorporates an ad-
anced set of desktop publishing fea-
ires to complement version 2.0
pabilities.
Professional Extension incorporates
panded Memory Support (**EMS**) for
er documents like books and techni-
nanuals, as well as densely packed
such as telephone directories and
gs. Different modes of automatic
al justification allow users to
ed space with

Gray-scale images, such
of pottery,

Профессиональное расширение пакета Ventura Publisher

Прежде чем начать издательскую деятельность на персональных компьютерах, необходимо принять решение, какой из издательских систем воспользоваться. Естественно, для того, чтобы начать такую работу, необходимо иметь соответствующую технику, а именно компьютер типа IBM PC/AT с оперативной памятью 640 Кбайт, жестким диском, графическим видеоадаптером, манипулятором мышь и операционной системой MS DOS версии не менее 2.1. Если вы будете работать с документами больших объемов, структуризованными определенным образом, то лучше воспользоваться такой системой, которая по максимуму автоматизирует процесс верстки. Для этих целей лучше подходит Ventura Publisher фирмы Xerox. Ventura Publisher позволяет осуществлять большое количество функций автоматически, что сильно сократит затраты времени при подготовке объемных документов, но при этом требует скрупулезной работы по предварительной разработке формата. Следует учесть, что системой Ventura лучше воспользоваться при выпуске большого количества изданий сходного формата (журналов, книг, технических описаний). По своим полиграфическим возможностям Ventura должна удовлетворить даже самых придирчивых любителей качественной печати. На PostScript-принтерах и принтере Laser Master пакет Ventura позволяет осуществлять поворот текста на 90, 180, 270 градусов. Также необходимо отметить, что Ventura имеет отдельную версию для локальной сети, и наконец, важно то, что на настоящий момент уже имеется большое число вспомогательных программ, расчитанных на совместную работу с пакетом Ventura, таких как Desktop Manager, Pubstar, VP Toolbox и XVP/Tabs.

Профессиональное расширение — что нового?

Итак, вы остановили свой выбор на пакете Ventura Publisher фирмы Xerox. Этот пакет, начиная с версии 2.0, кроме основной части, имеет специализированное дополнение, которое называется Профессиональное расширение (Professional Extension) и включает следующие основные возможности:
— поддержка расширенной памяти стандарта EMS;
— формульный набор;
— перекрестные ссылки, которые позволяют легко обновлять адресацию ссылок в любом месте документа;
— выключка по вертикали;
— генератор таблиц.

Профессиональное расширение требует дополнительно 600 Кбайт дискового пространства и 1.2 Мбайт для словаря. При работе с Профессиональным расширением вы можете применить все имеющиеся у вас навыки и знания по работе с системой Ventura 2.0, но их, естественно, будет недостаточно. Данная статья представляет собой попытку восполнить недостаток информации о некоторых особенностях применения Профессионального расширения. Следует отметить, что изложение материала рассчитано как на пользователей английской фирменной версии, так и на пользователей русифицированной версии пакета Ventura. Первоначально дается русский вариант названия режима, а в скобках — английский, далее в изложении при упоминании этого режима неоднократное количество раз дается только русский вариант названия.

Основы формульного набора

При работе с Профессиональным расширением вы можете ввести уравнение в любое место документа. Прежде всего необходимо установить автоматический интерлиньяж для тех абзацев, где будут размещаться уравнения. Для ввода уравнения выберите режим Текст (Text Editing), установите курсор в месте написания уравнения, выберите Вставить Спец. Параграф (Insert Special Item) в меню Редактирование (Edit). Из меню выберите режим Уравнение (Equation), и система предоставит вам экран для набора формул. Удерживая нажатыми клавиши CTR и C, вы получите меню с набором наиболее общих команд, создающих формулы. Существует более 100 команд и зарезервированных слов для написания формул и уравнений. Задав в верхней части экрана соответствующий синтаксис и подождав несколько секунд после ввода, вы увидите вашу формулу. Удерживая нажатыми клавиши CTR и D, вы выйдете из формульного редактора. Написанное вами уравнение появится в тексте.

В формульном языке есть 3 основные правила:

1. Необходимо вводить пробел до и после каждой команды и специальных слов.

2. Пробелы, размещаемые в уравнении, не печатаются (для задания пробела между символами используется знак ~ (тильда) — для нормального пробела и знак ˆ (крышка) — для минимального пробела).

3. Команды сразу же модифицируют выражение. При этом выражением считается любой символ, любая символьная строка или любая группа символов, неразделенных пробелами, которые помещены внутри фигурных скобок.

Например {xsup2~+~ysup2} дает x2+y2.

Перекрестные ссылки

Перекрестные ссылки — это ссылки к странице, рисунку, главе или номеру раздела. Наиболее простой вид перекрестной ссылки — это вставка номера текущей страницы или главы в любое место текста. Для этого необходимо:

— выбрать режим Текст;

— установить курсор в месте, где должна быть ссылка на номер страницы;

— выбрать режим Вставить Спец. Параграф в меню Редактирование;

— выбрать из меню Редактирование режим Ссылка (Reference), после чего появится диалоговое окно, где строку К Имени (At the Name) необходимо оставить пустой, а в строке Связь с (Anchor with) выбрать символ P для ввода номера текущей страницы, либо символ C для ввода номера главы.

Для создания ссылки, не относящейся к номеру страницы или главы, можно отмаркировать место в тексте, к которому относится эта ссылка. Маркировка текста осуществляется также, как вставка номера текущей страницы или главы, но только после нажатия клавиши Вставить Спец. Параграф в меню Редактирование необходимо выбрать режим Имя Маркера (Marker Name) и в появившемся диалоговом окне задать имя маркера.

Существует возможность присвоения метки связи не только определенному месту в тексте, но и окну. В дальнейшем имя метки связи используется как для привязки окна к тексту, так и для создания ссылки. Для маркировки окна необходимо:

— установить режим Окно (Frame);

— выбрать окно;

— в меню Окно выбрать режим Связь и Подписи (Anchors & Captions);

— ввести имя метки в строку Метка (Label).

При необходимости ввода перекрестной ссылки, вы опять проделываете все то же, что и при вставке номера текущей страницы или главы, и после выбора режима Вставить Спец. Параграф в меню Редактирование (режим Ссылка) необходимо ввести имя маркированного окна или имя метки текста в строку К Имени. Далее выбирается тип ссылки, который может принимать следующие значения:

— номер страницы P#;

— номер главы C#;

— рисунок F#;

— номер таблицы T#;

— номер раздела S* (ссылка с номером раздела, предшествовавшего месту, где стоит ссылка);

— текст с подписью C* (создается текст с подписью для заданного окна);

— текстовая переменная V* (произвольно задается текст ссылки).

Символ # обозначает задание значения нумерации по умолчанию (установленный для каждого типа нумерации). При необходимости имеется возможность ввода собственного значения типа ссылки. При подготовке изданий, которые состоят из нескольких глав, используется режим Связь Глав (Multi-chapter) в меню Сервис (Options), где можно собрать главы вашего документа и при этом обновить все перекрестные ссылки с помощью режима Перенумеровать (Renumber). Если имя маркера или метка ссылки не найдены, система выдает сообщение об ошибке и создает файл всех ненайденных ссылок. Эта информация очень полезна при корректировке перекрестных ссылок.

Из всего вышесказанного о перекрестных ссылках следует, что маркер можно установить либо в тексте, либо в выбранном окне и затем задать ссылку к этому маркеру в любом месте документа. Маркер и метка являются невидимыми. Когда вы вставляете перекрестную ссылку в текст и затем производите перенумерацию, появляется действительная ссылка.

Использование текстовых переменных в качестве перекрестных ссылок дает возможность модификации документов. Перекрестные ссылки такого типа удобно использовать для получения текущей даты, текущего номера издания, изменения названия продукции, получения адреса поставщика, заказчика и т.д. Это можно сделать и в текстовом редакторе, воспользовавшись процедурами поиска и замены, но при этом придется все время обновлять заменяемый текст. Ввод перемен-

ных позволит последовательно заменить требуемые значения. Рассмотрим пример: создадим переменную с именем Наименование, используя опцию Переменная (Variable) в режиме Вставить Спец. Параграф (меню Редактирование). Когда вы опишете эту переменную, введите имя: Продукция, затем текст в строку Текст Заменитель, например, Агрегат. Далее, в каждое место в издании, где вы хотите получить название продукции, вставьте Перекрестную Ссылку и укажите в строке К Имени: Наименование, а в строке Связь с: V*. Чтобы название продукции появилось в тексте, перенумеруйте издание, используя вариант Связь Глав. После этого слово Агрегат появится во всех местах, где вы вставляли перекрестную ссылку, связанную с именем переменной Продукция.

Выключка по вертикали

Выключка по вертикали обеспечивает размещение текста по всему пространству до нижней границы каждой колонки или полосы. Ее следует использовать для любого документа длиннее, чем страница. Выключка по вертикали автоматически добавляет пробелы сверху и снизу окон, таблиц, абзацев и, если это необходимо, между строками текста до тех пор, пока текст не достигнет нижнего края колонки или полосы. Пробелы добавляются в следующем порядке:

— между окнами и окружающим текстом до максимально разрешенного значения;

— между абзацами или между таблицами и абзацами до максимально разрешенного значения;

— между строками текста до максимально разрешенного для каждого абзаца значения.

Выключка по вертикали может быть задана в 4 режимах:
— Глава Полиграфическая (Chapter Typography);
— Окно Полиграфическое (Frame Typography);
— Абзац Полиграфический (Paragraph Typography);
— Вставить/Редактировать Таблицу (Insert/Edit Table).

Режимы Глава Полиграфическая и Окно Полиграфическое позволяют установить значение выключки по вертикали соответственно для главы или окна. В режиме Абзац Полиграфический можно задать максимальную величину отбивки абзаца сверху и снизу, а также максимальное значение интерлиньяжа, а режим Вставить/Редактировать Таблицу позволяет задать максимальное значение отбивки сверху и снизу таблицы. Для большинства документов, как правило, устанавливают следующие параметры из режима Глава Полиграфическая:
— Выключку по Вертикали Внутри Окна (Vert. Just. within Frame) — на значение Шпон (Carding);
— Выключку по Вертикали Вокруг Окна (Vert. Just. around Frame) — на значение Нефиксированная (Moveable).

Остальные параметры, входящие по умолчанию в каждый стилевой файл, устанавливают следующие параметры:

В режиме Глава Полиграфическая:
— Выключка по Вертикали Разрешена (Vert. Just. Allowed) — на значение 100%;
— Отбивка Сверху Окна (At Top of Frame) и Отбивка Снизу Окна (At Bottom of Frame) (Опция Глава Полиграфические Параметры) — на значение интерлиньяжа Основного текста (body text).
В режиме Абзац Полиграфич.:
— Сверху Абзаца (At Top of Para) и Снизу Абзаца (At Bottom of Para) (Полиграфические Параметры) установлена равными Сверху и Снизу абзаца Основного текста (body text) (опция Отбивка Абзаца (Spacing))
— Между Строками Абзаца (Between Lines of Para) (опция Абзац Полиграфические Параметры) устанавливается на ноль. Если вы хотите использовать эту опцию, задайте около 10% интерлиньяжа Основного текста (body text). Для дескрипторов (tag) с большим кеглем, например, для заголовков, задайте больший процент относительно интерлиньяжа данного дескриптора.

Если Основной текст в смежных колонках должен иметь совпадение по строкам, активизируйте Отбивка для Выключки по Вертикали Внутри Окна (Vert. Just. within Frame) и установите все значения выключки по вертикали во всех 4-х перечисленных выше опциях кратными интерлиньяжу Основного текста.

Генератор таблиц

Таблицей называют любой текст, размещенный в виде матрицы с рядами и колонками. Графа — это элемент пересечения ряда и колонки. Таблицу можно вставить между двумя любыми абзацами. Чтобы создать таблицу, необходимо:

1. Перейти в режим Вставить/Редактировать Таблицу;
2. Поместить курсор в место между абзацами, где будет располагаться таблица;
3. Выбрать режим Вставить Таблицу, используя дополнительный селектор в левой стороне экрана.

Также существует возможность создать таблицу, находясь в режиме Текст. Для этого необходимо:

1. Установить режим Текст;
2. Поместить курсор текста в конец или начало абзаца, предшествующего таблице;
3. Выбрать Вставить Спец. Параграф в меню Редактирование;
4. Из меню выбрать режим Таблица (F9).

Итак, мы выяснили, как нам приступить к созданию таблицы. Предположим, что наша таблица будет выглядеть следующим образом:

название подразделения	ФИО сотрудника	должность	оклад	персональная надбавка
плановый отдел	Воронов	инженер	250	125
плановый отдел	Киселев	ст. инжен.	270	110
плановый отдел	Иванова	вед. экон.	300	100

Прежде всего определяется количество рядов и колонок. Для нашей таблицы — это *5* рядов и *5* колонок. Далее задается тип выравнивания. Если в нашем примере количество сотрудников будет большим, то надо допустить возможность разрыва таблицы на страницы. При переходе таблицы со страницы на страницу можно задать автоматическое повторение шапки, указав количество рядов, которые будут считаться шапкой таблицы.

Таблица может быть ограничена линейками различных модификаций. Также можно сделать невидимой горизонтальную и вертикальную сетку. При создании нашей таблицы у нас есть возможность установить такие параметры, как количество пробелов сверху таблицы, снизу таблицы, между строками, между столбцами и некоторые другие. В отличие от программ электронных таблиц типа Lotus 1-2-3 или SuperCalc, Профессиональное Расширение позволяет расширять графы таблицы для занесения данных большего размера, переносить текст в пределах одной графы, а также растягивать таблицу вниз. По мере увеличения текста, автоматически добавляются строки и увеличивается высота ряда (ввод текста необычайно удобен). В графе можно размещать только один абзац с текстом. Для ввода дополнительных строк с текстом используйте принудительный обрыв строки (одновременное нажатие клавиш CTR и Enter).

Чтобы модифицировать таблицу, необходимо выделить какую либо ее часть. Это делается в режиме Редактирования Таблиц, для чего курсор мыши устанавливается в левый верхний угол первой графы из того сегмента, который вы хотите выделить. Нажмите и удерживайте кнопку мыши нажатой. При этом перемещайте курсор в нижний правый угол последней графы выделяемого сегмента. После выделения требуемой части таблицы можно выполнять такие действия, как удаление, копирование, восстановление рядов и колонок, изменение любой графы в таблице, добавление колонок или рядов, изменение ширины колонки, присвоение графе дескриптора. Задавать ширину колонки вы можете двумя способами: выбрав колонку и установив Ширину Колонки из меню Редактирование или, удерживая нажатой клавишу ALT, поместите курсор мыши в середину изменяемой колонки и удерживая кнопку мыши, переместите правую границу колонки на новое место. Допустим, что в нашем примере в отдел принимают на работу нового сотрудника по фамилии Миронов и нам надо добавить строку в нашу таблицу. Чтобы вставить строку в конкретном месте, необходимо выделить ту часть таблицы (строку, графу, группу строк), перед которой мы будем вставлять строку. Мы хотим вставить нового сотрудника после фамилии Воронов. Выделяем необходимую часть таблицы, при этом система требует подтверждения: “Вы уверены, что хотите вставить 1 Ряд(ы) перед рядом #4 в вашей таблице?“. После подтверждения указанного действия и ввода новой строки, наша таблица приобретет следующий вид:

название подразделения	ФИО сотрудника	должность	оклад	персональная надбавка
плановый отдел	Воронов	инженер	250	125
плановый отдел	Миронова	лаборант	150	50
плановый отдел	Киселев	ст. инжен.	270	110
плановый отдел	Иванова	вед. экон.	300	100

В большинстве случаев графы бывают различных размеров. Часто в самой таблице бывает несколько колонок с одной общей шапкой, например выпуск продукции за различные временные интервалы.

вид продукции	выпуск продукции тыс.тонн					
	1985	1986	1987	1988	1989	1990

Чтобы сделать такую шапку, необходимо создать табличку из 7 колонок и n-го количества рядов, выделить вместе 2, 3, 4, *5*, 6 и 7 колонки и затем выбрать команду Соединить графы. Таким образом, возможность соединения и разделения граф является очень полезной и наиболее часто используемой при создании сложных комплексных таблиц.

Первоначально система присваивает всем графам один дескриптор с именем Table Text. Далее каждой графе можно присвоить свой дескриптор. Однако лишь несколько опций меню Абзац полезны в таблицах, поэтому лучше минимизировать количество дескрипторов и использовать их только для задания различных выключек. Шрифт в графе лучше изменять в режиме Текст. Для увеличения расстояний между строками в графе используйте принудительный обрыв строки.

Стоит несколько слов сказать о задании параметра Автозаполнителя (Auto-Leader) в меню Параметры Табуляции (Tab Setting) (меню Абзац). Это позволяет автоматически заполнить графу любым символом, например, тире. Чтобы использовать этот режим, необходимо установить параметр Вкл. (On) в строке Автозаполнитель, а затем установить требуемые значения параметров Символ Заполнитель (Leader Char) и Межсимвольный пробел (Leader Spacing). Для ограничения ширины заполнения графы используйте Втяжку Слева (In From Left)/Справа (In From Right). Если дескриптор присвоен пустой графе с текстом, то символ заполнит промежуток от конца последней строки до правого поля графы. Для этих целей не нужно вводить символ табуляции.

Табличный конвертор

Таблицы могут содержать как текст, так и различные числовые данные, которые бывает необходимо предварительно обработать и выдать результаты. Система Ventura не имеет функций обработки табличных данных, т.к. ее основное предназначение — это изда-

тельская деятельность. В этом случае необходимо импортировать в систему Ventura табличные данные, создаваемые в других системах.

Поскольку в системе Ventura 2.0 не было средств для построения таблиц, то и не было возможности импорта файлов системы dbase, Lotus 1-2-3 и даже файлов с расширением PRN (предназначенных для непосредственного вывода на устройство печати).

Профессиональное расширение имеет конвертор, который может автоматически распознавать табличные данные и преобразовывать их в формат системы Ventura. Для этого можно использовать вариант PRN в Таблицу в режиме Загрузить Текст/Рисунок (Load Text/Picture).

Следует помнить, что прежде, чем импортировать файлы с табличными данными в систему Ventura, их необходимо сформировать в соответствующих пакетах в формате файлов для печати. Так, для пакета Lotus 1-2-3 это будет файл с расширением PRN, а для пакета dbase — файл с расширением TXT.

К сожалению, генератор таблиц Профессионального расширения не воспринимает файлы с расширением .WKS или .WK1 пакета Lotus 1-2-3, а также файлы DBF пакета dbase — обычные источники табличных данных. В этом случае можно применить утилиту XVP/TABS.

Используя эту утилиту, вам не придется скурпулезно форматировать вашу электроннную таблицу для преобразования в ASCII-файл. Очень важно то, что XVP/TABS автоматически считает все позиции табуляции, записывает их в стилевой файл и присваивает общие дискрипторы для быстрого форматирования таблиц. XVP/TABS облегчает процесс оформления таблицы, автоматизируя многие операции с метками строк, шапкой таблицы и т.д., а также обеспечивает верное форматирование перекрестных ссылок.

И.Изосимова

По материалам:

S.Roth “Ventura Turns Two“, PC World, February, 1989.
“101 Hot Tips“, Publish!, June, 1989.
“Page Makeup & Design“, Publish!, June, 1989.
S.Cummings, R.Eckhardt, D.Will-Harris “The Twopenny Toolbox“, Publish!, August, 1989.
R.Jantz “Variety of Venturas Debut This Spring“, Publish!, February, 1990.
“Feel the Power“, Publish! April, 1990.

PostScript-библиотека на компакт-диске

Фирма Agfa впервые выпустила свою обширную коллекцию PostScript-шрифтов на компакт-диске. Диск содержит шрифты разработки фирм Adobe и Agfa.

Собственно диск будет поставляться за номинальную цену, а шрифты — за цену, примерно на 10% меньшую, чем при поставке на флоппи-дисках. Шрифты хранятся на диске в закодированном виде, а пользователь при покупке будет получать пароль для обработки и копирования на свой винчестер только тех шрифтов, за которые он заплатил.

Большим преимуществом новой системы может считаться то, что пользователь сможет получать доступ к новым шрифтам путем приобретения новых паролей к информации на диске.

В комплект поставки входит также каталог всех шрифтов и информация о том, где и как можно заказать новые.

Desktop Publishing Today, May 1990

Вторая версия (Level 2) языка описания страниц PostScript фирмы Adobe.

Вторая версия включает в себя новые графические и текстовые операторы для улучшения функциональных свойств.

Введенный механизм сжатия данных и изображений уменьшит затраты времени на передачу информации и объем памяти, необходимый для хранения PostScript-файлов, содержащих изображения. Улучшены также и все основные режимы; появился новый улучшенный алгоритм наложения типографского растра на шрифт. Добавлены и новые режимы, в том числе режим печати заполненных бланков, позволяющий хранить в памяти принтера изображение бланка и добавлять к нему лишь изменяющиеся поля данных.

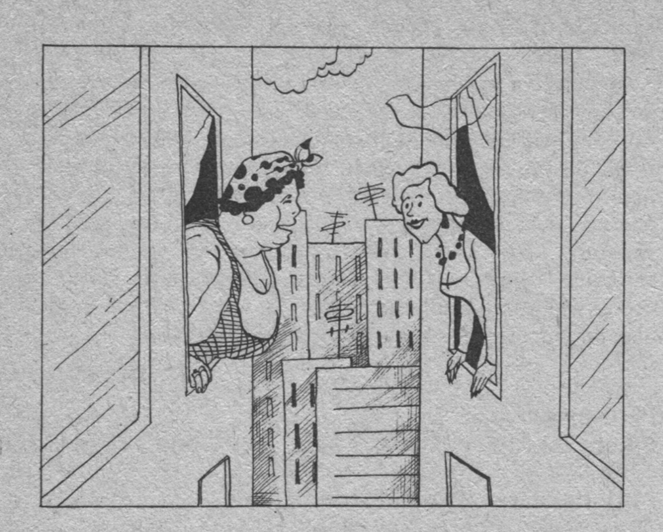
Что касается цветов, где недостатки PostScript были наиболее очевидны, фирма Adobe применила новую аппаратно-независимую модель для их описания. Результатом этого должно быть получение одинаковых цветных изображений даже на принципиально различных устройствах вывода, что может оказаться полезным тем, кто использует PostScript для полиграфического цветоделения.

В PostScript level 2 встроена и поддержка специфичных для конкретного принтера характеристик, таких как размер бумаги и различные устройства ее подачи. Кроме того, будет облегчена работа с памятью, цветами, составными шрифтами, улучшен дисплейный вариант языка.

Фирма заявляет, что новая версия языка будет полностью совместима с нынешней, а компания будет выпускать новые программные драйверы. Они дадут возможность использовать все преимущества новой версии Macintosh, Windows version 3.0 и OS/2.

Технические изменения фирма будет производить вместе с компаниями, купившими лицензии на PostScript. По сообщениям фирмы, первые устройства, поддерживающие PostScript level 2, появятся не ранее начала 1991 года.

Desktop Magazine, October, 1990



Наверное, каждому программисту, особенно начинающему, хочется, чтобы его программы выглядели профессионально. Любая профессионально написанная программа отличается развитым интерфейсом пользователя, включающим в себя многоуровневую систему пользовательских меню, выполненных обычно в виде распахивающихся окон, различные диалоговые окна и средства диагностики. Предлагаемая вашему вниманию библиотека программ содержит достаточно полный набор функций, необходимых для решения таких задач. Статья предназначена в основном для тех, кто только начинает программировать на ассемблере и хочет научиться сопрягать модули, написанные на ассемблере, с языками высокого уровня.

КАК СОЗДАТЬ ОКОННЫЙ ИНТЕРФЕЙС

При разработке оконного интерфейса программист обычно сталкивается с рядом проблем, которые не слишком просто решаются при помощи языков высокого уровня. Здесь возможны два варианта решений. Можно воспользоваться стандартными библиотечными функциями, имеющимися в данной реализации языка. В этом случае модуль оконного интерфейса обычно получается достаточно большим по объему (что не оправдывает его включения в небольшую программу) и работающим относительно медленно.

Второй вариант решения проблемы заключается в том, чтобы осуществлять вывод изображения непосредственно в видеопамять, минуя стандартные функции вывода (как это и делается в большинстве профессиональных программ). Языки высокого уровня обычно позволяют решать подобную задачу, но при этом также имеется ряд недостатков. Во-первых, для осуществления тех функций (например, по пересылке данных), которые можно было бы целиком выполнить на внутренних регистрах процессора, приходится вводить довольно много переменных, которые размещаются в памяти. Компилятор зачастую не может создать такой модуль, в котором целиком использовались бы регистровые переменные. То есть, при многочисленных пересылках данных будут производиться также и многочисленные излишние обращения к памяти, что значительно замедлит весь процесс. Во-вторых, при работе с большими моделями памяти (Compact, Large, Huge), в которых данные могут размещаться по всему объему памяти и имеют длинные (far) адреса — а при обращениях к видеопамяти всегда приходится использовать far-адреса — работа с такими адресами опять-таки усложнит и замедлит ход выполнения вашей программы.

Большинства этих проблем поможет избежать ассемблер. Здесь вы можете максимально использовать регистровые переменные, а при работе с far-адресами

сегментный адрес обычно приходится вычислять только один раз, что позволяет значительно ускорить всю пересылку, и, наконец, можно оптимизировать вашу программу до действительно минимального объема и максимальной скорости выполнения. Отказываться от языков высокого уровня, однако, тоже не следует. Здесь нужно найти разумный компромисс между тем, какие модули проще и удобнее выполнять на ассемблере, а какие — на языках высокого уровня.

Особенности использования встроенного ассемблера

Предлагаемая библиотека функций оконного интерфейса была разработана для программы предсказания элементов вторичной структуры белков по их аминокислотной последовательности по методу Н.Морозова. Программа выполняла следующие операции, типичные для большинства прикладных программ: чтение данных из файла или файлов, обработку данных и запись результатов в выходной файл. При этом можно было изменять условия задачи при помощи ключей, имеющихся в меню. Ниже приводится пример простой программы, осуществляющей аналогичные действия.

Все функции были разработаны при помощи программных средств, предоставляемых пакетом Turbo C Professional фирмы Borland International (реализации Turbo C 2.0 или Turbo C++ 1.0), но вполне могут быть перенесены и на любой другой вариант Си или другого языка высокого уровня. Часть функций выполнена на Си, часть — на ассемблере. Для разработки последних в основном использовался встроенный ассемблер пакета Turbo C (inline assembler), либо они впоследствии были адаптированы к встроенному ассемблеру.

Многие программисты избегают пользоваться встроенным ассемблером Turbo C (равно как и других пакетов Си) по причине, видимо, недостаточно ясного представления о его возможностях. Встроенный ассемблер позволяет программисту избавиться от многих хлопот, связанных со стыковкой модулей, написанных на Си, и модулей, написанных на ассемблере. В пользу его применения говорят следующие аргументы. Во-первых, можно пользоваться стандартным способом описания функций, принятым в Си; при этом не нужно заботиться ни об управлении сегментами, ни об объявлении процедур, ни о соответствии моделей памяти, т.к. все это настраивается автоматически во время компиляции. Во-вторых, встроенный ассемблер обеспечивает полный доступ ко всем константам, переменным, элементам структур и объединений и даже функциям Си посредством соглашений, принятых в Си. Удобство доступа к переменным Си во встроенном ассемблере является одной из главных причин целесообразности применения встроенного ассемблера при разработке смешанных Си и ассемблерных программ. В-третьих, Turbo C заботится о многих программных деталях, как например, вход и выход из функций, передача параметров и размещение внешних и локальных переменных. В-четвертых, можно использовать смешанный код Си и ассемблера в одном и том же модуле (например, производить вызов функций обычным способом, принятым в Си), правда, пользоваться этим нужно крайне осторожно. И, наконец, не последнюю роль может играть тот факт, что фирма Borland International в своих разработках сама широко использует встроенный ассемблер, и многие библиотечные функции Turbo C также написаны на нем. В любом случае для начинающего (да и не только начинающего) программиста встроенный ассемблер предоставляет прекрасную возможность создавать модули, которые будут работать в составе любой программы, и которые не нужно будет каждый раз настраивать под определенную модель памяти. Конечно, классический ассемблер также предоставляет подобную возможность, но там это осуществляется значительно более сложными способами.

Существуют, правда, некоторые ограничения, которые нужно иметь в виду при составлении программ на встроенном ассемблере, а также некоторые правила, которые отличаются от правил написания программ на чистом ассемблере.

Так модули, написанные с использованием встроенного ассемблера, желательно помещать в отдельный от остальных Си-модулей файл (или файлы), так как компилятор Turbo C, встретив в файле код встроенного ассемблера, останавливает компиляцию, возвращается к началу файла и затем обрабатывает этот файл заново, но уже отключив всякую оптимизацию. Эта мера является вполне оправданной, т.к. делается предположение, что программист, используя код встроенного ассемблера, имеет ясное представление о том, как работает тот или иной участок программы и преследует определенную цель по улучшению ее работы, чему оптимизация может только помешать. Компилятор Turbo C предоставляет возможность получить полный ассемблерный листинг вырабатываемого кода, чтобы программист мог более детально разобраться в работе своей программы. Для этого нужно воспользоваться той версией компилятора, которая работает с командной строкой (tcc.exe), с ключом -S.

Для ускорения процесса компиляции файлов, содержащих код встроенного ассемблера, используют либо опцию -B в командной строке tcc.exe, либо в начало файла помещают директиву препроцессора #pragma inline , которая оповещает компилятор о том, что нужно вначале сгенерировать ассемблерный код, а затем вызвать Turbo Assembler для его трансляции. Нужно иметь в виду, что интегрированный интерфейс пользователя компилятора Turbo C 2.0 не поддерживает код встроенного ассемблера, поэтому для этой версии приходится проводить раздельную компиляцию, а в проект включать уже объектные модули. Версия компилятора Turbo C++ 1.0 поддерживает код встроенного ассемблера в интегрированном окружении.

Каждая инструкция или директива встроенного ассемблера должна начинаться с ключевого слова asm и

3 Зак. 2125

заканчиваться точкой с запятой или последовательностью CR-LF (в реализации Turbo C++ 1.0 предусмотрена возможность объединения инструкций встроенного ассемблера в блоки с использованием конструкции asm { ... }). Другое правило касается комментариев, которые должны соответствовать соглашениям, принятым в Си (а не в ассемблере!). Это разумно, если принять во внимание тот факт, что файл вначале обрабатывается компилятором Си, а уже потом — ассемблером.

Ограничения, налагаемые встроенным ассемблером, касаются, в основном, использования меток. Все метки, на которые производятся ссылки в инструкциях условных и безусловных переходов (jumps), должны быть метками Си, остальные же метки должны быть метками ассемблера и предваряться ключевым словом asm (например: asm Mybyte db 0;). Для последнего случая компилятор Turbo C++ 1.0 выдает предупреждение о том, что в строке была обнаружена неопознанная ассемблерная инструкция, так что программист должен быть более внимательным и аккуратным при обращении с метками.

Другие соглашения относятся уже не только ко встроенному ассемблеру и определяют правила использования внутренних регистров процессора 80i86, порядок передачи параметров и возврата значений функциями, а также типы и размер данных в стандарте Си.

Использование регистров

Стандарт Turbo C налагает следующие ограничения на использование регистров ассемблерными функциями. При возврате из функции регистры BP, SP, CS, DS и SS должны содержать те же значения, что и при входе в функцию. Регистры AX, BX, CX, DX, ES и флаги могут быть изменены произвольно.

Регистры SI и DI представляют собой особый случай, так как они используются Turbo C для хранения регистровых переменных. Если использование регистровых переменных разрешено в том Си-модуле, из которого производится вызов вашей ассемблерной функции, то значения SI и DI должны быть сохранены при выходе из функции; если регистровые переменные не используются — значения SI и DI могут изменяться без ограничений. Однако, чтобы иметь уверенность в том, что ваша функция будет работать в составе любой программы, целесообразно всегда сохранять значения регистров SI и DI.

При использовании встроенного ассемблера отпадает необходимость заботиться о сохранении регистров SI и DI, так как в том случае, если разрешено использовать регистровые переменные и компилятор Turbo C находит в вашей программе какие-либо обращения к регистрам SI и DI, он автоматически генерирует код, сохраняющий эти регистры и восстанавливающий их при возврате из функции. (Здесь, правда, имеется один маленький недостаток. Компилятор не распознает, какие обращения к регистрам изменяют их содержимое, а какие нет. Поэтому, в случае, например, просто чтения содержимого регистра — mov ax,si — компилятор все равно будет генерировать сохраняющий код.)

Типы используемых данных

В этой таблице приводятся соответствия между типами данных Си и ассемблера.

Типы данных Си	Типы данных Ассемблера
unsigned char	byte
char	byte
enum	word
unsigned short	word
short	word
unsigned int	word
int	word
unsigned long	dword
long	dword
float	dword
double	qword
long double	tbyte
near *	word
far *	dword

Порядок передачи параметров и возврата значений ассемблерными функциями

Для всех дальнейших рассуждений, касающихся стека, условимся, что основание стека располагается вверху (у верхней границы сегмента SS), а вершина стека — внизу (на нее указывает SP), то есть так, как это и организовано в памяти компьютера. После загрузки программы в память указатель стека SP устанавливается на верхнюю границу сегмента SS. При засылке какого-либо слова в стек, значение указателя стека вначале уменьшается на 2, и затем указанная величина заносится в память по адресу SS:SP.

Turbo C передает параметры своим функциям через стек. Перед вызовом функции Turbo C засылает все значения, указанные в списке передаваемых параметров в стек, начиная с последнего (правого) и кончая первым (левым). Далее в стек засылается адрес возврата из функции, который, в зависимости от модели памяти, может быть типа near или far и иметь размер, соответственно, word или dword. В случае far-вызова в стек вначале засылается адрес сегмента вызывающей функции, а затем — смещение. Таким образом, сразу после вхождения в функцию переданные ей параметры, начиная с первого, будут размещены в стеке, с адреса SP+2 и выше (для near-вызова), либо SP+4 и выше (для far-вызова).

Для получения доступа к переданным параметрам в ассемблерных функциях используется стандартный прием (в этом нет необходимости при использовании встроенного ассемблера). Вначале сохраняется значение базового регистра BP в стеке, затем в него переда-

ется значение указателя стека. При возвращении из функции восстанавливаются значения SP и BP. В программе это выглядит так:

```
push bp
mov  bp,sp
.
.
.
mov  sp,bp
pop  bp
```

Таким образом, переданные параметры могут быть доступны через адресацию относительно базового регистра BP и будут расположены в стеке, начиная с адреса [BP+4] для near-вызова, либо с адреса [BP+6] для far-вызова — и выше.

Если в функции используются локальные переменные, то их обычно тоже размещают в стеке. Локальные переменные можно разместить также и в сегменте кода (CS), однако в этом случае обращение к ним будет осуществляться медленнее, т.к. понадобится использовать префикс переназначения сегмента (segment override prefix), и вычисление эффективного адреса будет происходить дольше (разумеется, в этом нет необходимости, если сегментные адреса CS и DS совпадают).

Для размещения локальных переменных в стеке резервируют место, вычитая из значения указателя стека SP совокупный размер всех локальных переменных. Например, для размещения трех локальных переменных типа word понадобится 3 слова, а указатель стека нужно будет уменьшить на 6:

```
sub sp,6
```

Это необходимо делать только в том случае, если в вашей функции имеются последующие засылки в стек, либо производятся вызовы подпрограмм.

Локальные переменные, таким образом, будут также доступны за счет адресации относительно базового регистра BP и расположены, начиная с адреса [BP-2] и ниже (для переменных типа word).

Внешние переменные размещаются в сегменте данных и адресуются обычным образом, т.е. либо относительно базового регистра BX, либо относительно индексных регистров SI и DI, либо непосредственным значением смещения.

Хочется подчеркнуть, что при использовании встроенного ассемблера Turbo C отпадает необходимость в применении таких сложных способов доступа к передаваемым параметрам и переменным (как локальным, так и внешним). Более того, если вы используете размещение данных в сегменте кода, встроенный ассемблер сам будет добавлять префикс переназначения сегмента CS: (segment override prefix) и генерировать правильный код.

Ваша ассемблерная функция может возвращать значения точно так же, как это делают Си-функции. При этом возвращаемые значения должны располагаться в следующих регистрах:

Тип возвращаемого значения	Размещение
unsigned char	AX
char	AX
enum	AX
unsigned short	AX
short	AX
unsigned int	AX
int	AX
unsigned long	DX:AX
long	DX:AX
float	8087 регистр вершины стека ST(0)
double	8087 регистр вершины стека ST(0)
long double	8087 регистр вершины стека ST(0)
near *	AX
far *	DX:AX

Обычно 8- и 16-битовые значения возвращаются в регистре AX, а 32-битовые значения — в регистрах DX:AX, со старшими 16 битами в регистре DX. Значения с плавающей точкой возвращаются в ST(0) — регистре вершины стека (top-of-stack или TOS) сопроцессора 8087, либо в том же регистре программного эмулятора сопроцессора, если таковой используется.

Со структурами дело обстоит несколько сложнее. Структуры длиной 1 или 2 байта возвращаются в регистре AX, а структуры длиной 4 байта возвращаются в регистрах DX:AX. Трехбайтовые структуры и структуры длиной более 4 байт должны быть помещены в сегменты статических данных, и затем указатель на соответствующую структуру должен быть возвращен вызывающей программе. Как и все другие указатели, указатели типа near возвращаются в AX, а указатели типа far — в DX:AX.

После возврата из функции стек должен быть восстановлен в то же состояние, что и до вхождения в нее, то есть к указателю стека нужно прибавить общую длину переданных функции параметров. Стандарт Си требует, чтобы этим занималась вызывающая программа. Поэтому вас не должно заботить восстановление стека после возврата из вашей функции, однако, если внутри вашей ассемблерной функции производятся вызовы других Си-функций, нужно обязательно после них восстанавливать стек. При использовании встроенного ассемблера Turbo C можно применить Си-код для вызова функций внутри вашей ассемблерной программы. В этом случае не нужно будет заботиться ни о передаче параметров функции, ни о восстановлении стека.

Необходимо также отметить, что стандарт Turbo C требует, чтобы все внешние метки в ассемблерных Си-функциях (имена функций и внешних переменных) начинались с символа подчеркивания `_'. Компилятор Turbo C автоматически добавляет начальные символы подчеркивания ко всем именам функций и внешних переменных, используемым в Си-коде вашей программы. Но если вы используете код чистого ассемблера для разработки ваших Си-функций, то необходимо следить за тем, чтобы все ссылки на функции Turbo C

и внешние переменные начинались с символов подчеркивания; кроме того, все имена ваших ассемблерных функций и тех переменных, которые объявлены общедоступными (public), также должны соответствовать этому правилу.

Создание универсальных ассемблерных функций

Встроенный ассемблер дает пользователю возможность создавать универсальные ассемблерные функции, которые могут работать в составе любой программы и с любой моделью памяти. Все, что для этого нужно сделать — это лишь объявить все указатели, передаваемые функции в качестве параметров или возвращаемые функцией, указателями far-типа и в дальнейшем манипулировать ими соответствующим образом. Обо всем остальном позаботится компилятор Turbo C.

Работа с far-адресами лишь ненамного замедлит ход выполнения вашей функции, т.к. для обработки строк в ассемблерной программе адрес сегмента обычно требуется определить лишь один раз. Функция же в результате получит гибкость и мобильность, и вам не нужно будет заботиться о ее переносимости в другие программы.

К сказанному остается лишь добавить, что если для разработки своих функций вы пользуетесь встроенным ассемблером Turbo C, то всегда желательно посмотреть, как же в действительности будет выглядеть ваша программа после обработки ее компилятором. Это особенно важно в случае, когда в программе присутствует смешанный код Си и ассемблера. Компилятор tcc.exe с ключом -S позволит вам получить ассемблерный текст вашей программы. При желании вы можете в дальнейшем взять этот текст за основу и работать уже с чистым ассемблером, — в любом случае такой подход поможет лучше понять, как взаимодействуют друг с другом функции Си и ассемблера.

Описание библиотеки

Предлагаемый вашему вниманию набор функций позволяет строить на экране окна пользовательских меню, производить выбор нужной позиции в окне (при помощи курсора или нажатия "горячей" клавиши) и соответствующим образом реагировать на выбор; строить диалоговые окна и редактировать строку в диалоговом окне, а также выдавать некоторую диагностику. Помимо этого, в состав пользовательского меню на каждом уровне вложения можно ввести набор дополнительных функциональных клавиш, которые не будут непосредственно присутствовать в окне меню, но которые также будут вызывать в программе необходимый отклик. К достоинствам библиотеки относится также и то, что функции автоматически отслеживают переключение страниц на экране.

Сразу оговоримся, что библиотека не была специально предназначена для работы с видеоадаптером CGA, как наименее популярным и, видимо, потерявшим свою актуальность на современном этапе развития аппаратных компьютерных средств. В видеоадаптере CGA не предусмотрено аппаратное управление распределением времени доступа к видеопамяти между хозяином системной шины (например, центральным процессором) и непосредственно контроллером дисплея. В результате, при прямых обращениях к видеопамяти в адаптере CGA могут возникать помехи изображению, проявляющиеся в виде "снега" на экране.

Для правильной работы с видеоадаптером CGA тех ассемблерных функций, которые осуществляют прямые обращения к видеопамяти, в них необходимо ввести цикл ожидания обратного хода луча по вертикали (vertical retrace) до каких бы то ни было обращений к видеопамяти и затем выключить луч дисплея. После вывода информации в видеопамять нужно снова включить луч. Последняя процедура осложняется тем, что регистр управления режимами видеоадаптера CGA рассчитан только на запись информации (write only), поэтому приходится проверять текущий режим работы дисплея, чтобы произвести правильное переключение. Само собой разумеется, что такие добавления существенно замедлят ход выполнения функции. Ассемблерные процедуры отключения и включения дисплея CGA могут выглядеть приблизительно следующим образом:

```
_CGA_off  proc  far
          ; прочитать порт состояния CGA
          mov   dx,3DAh
          ; ждать обратного хода луча по
          ; вертикали
again:    in    al,dx
          test  al,8
          ; если бит 3 не установлен, то
          ; возврат в цикл
          jz    again
          mov   dx,3D8h
          ; выключить луч, очистив бит 3 в
          ; регистре управления режимами CGA
          mov   al,00100101b
          out   dx,al
          ret
_CGA_off  endp

_CGA_on   proc  far
          xor   ax,ax
          mov   es,ax
          ; регистр управления режимами CGA
          mov   dx,3D8h
          ; значение для режима 2
          mov   al,00101101b
          ; определить режим дисплея
          cmp   byte ptr es:[449h],2
          je    skip_mode3
          ; значение для режима 3
          mov   al,00101001b
skip_mode3:
          ; включить луч
          out   dx,al
          ret
_CGA_on   endp
```

Помимо всего сказанного, с видеоадаптером CGA не будет работать функция toggle_intensity_blinking(), т.к. переключение бита мигания/интенсивности в CGA производится другим способом — через регистр управления режимами. Вызов этой функции приведет к "зависанию" компьютера. И, наконец, для работы библиотеки с видеоадаптером CGA нужно будет подобрать подходящие цвета для окон, курсоров и т.д.

С видеоадаптерами MDA, EGA, VGA библиотека работает должным образом.

Некоторые из функций, представленных в библиотеке, снабжены лишь небольшим количеством комментариев вследствие того, что в них производятся многочисленные манипуляции данными, описание которых заняло бы слишком много места и не было бы сколько-нибудь интересным для читателя. Тем же, кто захочет более детально разобраться в работе таких функций с целью, возможно, дальнейшего их совершенствования, предлагаем воспользоваться отладчиком (например, Turbo Debugger) и проработать программу по шагам.

Помимо этого, некоторые из используемых функций могут показаться кому-нибудь излишними, либо отчасти повторяющими библиотечные функции Turbo C, но данный вариант библиотеки оконного интерфейса является всего лишь авторской версией, не претендующей на безоговорочное признание и оставляющей широкий простор для собственной изобретательности искушенного читателя.

Библиотека позволяет строить окна в ставшем популярным в последние годы стиле — с отбрасываемой тенью, что создает ощущение некоторой объемности экрана. Тень при этом не подавляет полностью символы, находящиеся под ней на экране, а лишь меняет их атрибуты. При этом, правда, возникает одна небольшая проблема. На что должно отбрасывать тень первое окно? Ведь на пустом черном экране тень не будет видна, либо придется изменить ее цвет, что будет, в общем-то, противоестественным. Для разрешения этой задачи в профессиональных программах обычно "закрашивают" экран каким-либо цветом, на фоне которого отбрасываемая тень будет хорошо видна. Такой способ решения, однако, сразу же порождает множество других проблем.

Поэтому в своей программе мы предлагаем сделать для первого окна темно-серую тень (что выглядит довольно естественным, в отличие от других цветов). Однако, темно-серый цвет не входит в палитру стандартных цветов для фоновых атрибутов символа, т.к. 15-й бит в байте атрибутов обычно отвечает за мигание символа на экране. Эта проблема, между тем, легко разрешима, если переключить бит мигания/интенсивности регистра управления режимами атрибутов в контроллере атрибутов EGA/VGA в состояние разрешения фоновой интенсивности (0), что и осуществляется при помощи функции **toggle_intensity_blinking()**. В результате появляется возможность получения на экране темно-серой тени, которая и используется при построении первого окна.

В качестве примера использования функций оконного интерфейса предлагаем вам вариант простой программы, разбивающей текстовый файл на страницы и устанавливающей поля для четных и нечетных страниц. Программа производит чтение информации из файла (или файлов по заданному шаблону), обработку текста в соответствии с заданными условиями и запись результатов в указанный файл. Для такой программы интерфейс пользователя получается одноуровневым (т.е. без вложенных меню), но вы в ваших разработках можете легко получить любое количество уровней вложения.

Проект программы MAKEPRT.PRJ должен включать в себя следующие файлы: MAIN.C, UTILIT1.C, UTILIT2.C, UTILIT3.C, INLUTIL.C (либо, в случае Turbo C 2.0, — MAIN.C, UTILIT1.C, UTILIT2.OBJ, UTILIT3.OBJ, INLUTIL.OBJ. Последние файлы содержат код встроенного ассемблера, и они должны быть заранее оттранслированы при помощи tcc.exe).

Программа начинается с установки альтернативной функции обработки прерывания 23h (Ctrl-Break DOS handler) break_control(). Эта функция позволяет производить экстренное прерывание процесса обработки информации и возвращает управление в окно меню пользователя после того, как произошло прерывание 1Bh (Ctrl-Break). Таким образом, пользователь получает возможность осуществления более полного контроля над работой программы, а сама программа становится более защищенной и гибкой в реагировании на непредусмотренную ситуацию.

Затем при помощи функции **break_off()** временно отключается обработка прерывания 1Bh (Ctrl-Break) и запоминается старый вектор этого прерывания. Это делается по следующим соображениям. Обычно, при обработке прерывания Ctrl-Break устанавливается внутренний флаг DOS, который сигнализирует о том, что прерывание Ctrl-Break имело место. Далее DOS периодически проверяет состояние внутреннего флага (это производится обычно во время операций ввода-вывода, в зависимости от состояния другого флага — Ctrl-Break Check). Если во время проверки флаг оказывается установленным, то он сбрасывается и производится вызов прерывания 23h (адрес выхода по прерыванию Ctrl-Break). Вектор именно этого прерывания и перехватывается функцией break_control(), о которой шла речь выше. Таким образом, если во время работы пользователя с окнами меню будет вызвано прерывание Ctrl-Break (1Bh), в работе программы не произойдет никаких немедленных изменений. Зато в дальнейшем, при обращениях к функциям ввода-вывода DOS, сразу же будет вызвана программа обработки прерывания 23h (break_control()), и будет произведен преждевременный выход из функции обработки информации. Вот почему целесообразно включать программу обработки прерывания Ctrl-Break (1Bh) только на время обработки информации, в остальное же время будет удобно отключать эту программу. Само собой разумеется, что перед выходом из прикладной прог-

раммы необходимо восстановить значение старого вектора обработки прерывания Ctrl-Break.

Далее в программе MAKEPRT производится чтение текущего видеорежима и формы мигающего курсора, которая затем будет использована для курсора в диалоговом окне, а сам курсор временно прячется. Если видеоадаптер находится в монохромном режиме, производится смена атрибутов окон и курсоров меню.

Затем производится заполнение атрибутами массивов курсоров меню, включение окна главного меню и обработка значений выбора, полученных в результате реакции интерфейса на действия пользователя.

В качестве примера использования дополнительных функциональных клавиш, не входящих в окно меню, в программе показана реакция на клавишу ESCAPE (которая осуществляет выход из программы) и на клавишу Ctrl-O (которая временно гасит окно меню, чтобы можно было увидеть информацию на экране).

Подпрограммы включения и выключения окна главного меню и диалоговых окон, а также подпрограммы редактирования имен файлов и численных значений и подпрограммы переключения опций, — выполнены в виде самостоятельных функций, основные параметры которых определены на внешнем уровне и являются универсальными для любых значений выбора в окне главного меню.

Функция редактирования имени файла **edit_fname()** производит редактирование строки в диалоговом окне и возвращает полученное имя файла в указанном в качестве входного параметра буфере.

Функция редактирования целых чисел **edit_number()** производит редактирование числа непосредственно в окне главного меню и возвращает результат по адресу, указанному в переданных параметрах. При этом осуществляется отслеживание возможных ошибок ввода.

Функция переключения опции **toggle_switch()** производит переключение ключа, переданного в качестве входного параметра, на противоположное значение, а также выводит в окно меню индикатор текущего состояния ключа.

Функция **process()** осуществляет обработку информации (разбиение файлов на страницы) в соответствии с заданными условиями. Предусмотрена возможность отключения генерации номеров страниц в выходном потоке, а также отключения выдачи текста на экран, что значительно ускоряет процесс обработки. Перед вызовом функции process() производится восстановление старого вектора прерывания 1Bh (Ctrl-Break) при помощи функции break_on(), выключается окно меню, восстанавливается форма курсора, а также устанавливается адрес перехода для функции longjmp(). После возврата из функции process() выполняются обратные действия.

Функция **break_control()** закрывает все открытые потоки и, если не было ошибки, передает управление на адрес возврата, заранее установленный функцией setjmp() (т.е. в окно главного меню); в противном случае восстанавливается бит мигания/интенсивности (если это необходимо) и производится выход из программы.

Далее следует описание функций, входящих непосредственно в библиотеку оконного интерфейса.

Функция get_pathname()

Функция является дополнением к библиотечным функциям Turbo C findfirst(), findnext() и searchpath(). Она позволяет получить имя и маршрут поиска файла в соответствии с указанной моделью поиска и именем файла, найденным по этой модели при помощи функций findfirst() или findnext(). Полученное имя в дальнейшем может быть передано в качестве входного параметра для функции searchpath(), которой нельзя напрямую передавать имя файла, найденное функциями findfirst() и findnext(), т.к. обнаруженный файл может не содержаться в каталогах, указанных в стандартном пути поиска, установленном в переменной PATH окружения DOS.

Функция shrink_fname()

Функция производит "сжатие" переданного имени файла с тем, чтобы его в дальнейшем можно было отобразить в окне меню. Сжатие производится в соответствии с соглашениями, принятыми в DOS. При невозможности указания полного пути поиска файла, он заменяется в выходной строке многоточием. Помимо этого функция добавляет к переданному имени файла стандартное расширение (если таковое отсутствовало в имени). В качестве стандартного расширения функции может быть передан указатель на нулевую строку.

Функция wildcard_mes()

Функция производит выдачу на экран окна с предупреждением о наличии в модели имени выходного файла недопустимых символов *,? и ждет нажатия клавиши ESCAPE, чтобы погасить окно. В качестве входного параметра функции передается только номер строки экрана, соответствующей верхней рамке окна. По горизонтали окно размещается в центре экрана. Если вы хотите иметь полный контроль над расположением окна на экране, необходимо соответствующим образом модифицировать функцию. Функция не отслеживает ошибок, связанных с передачей ей координат, выходящих за рамки экрана — этим должна заниматься вызывающая программа.

Функция fexists_mes()

Эта функция строит на экране окно с предупреждением о том, что указанный выходной файл уже имеется на диске и возвращает режим открытия этого файла: перезапись файла, прибавление к концу существующего файла, либо отмена намеченной операции. Функция позволяет окну автоматически подстраивать свои горизонтальные размеры под длину переданной строки с именем файла, правда, делается это в ограниченных пределах. Строка имени файла с длиной,

превышающей максимально установленную для окна, сжимается в соответствии с соглашениями DOS. Возвращаемые значения выбираются либо при помощи бруска курсора, управляемого стрелками, либо при помощи "горячих" клавиш. По горизонтали окно размещается в центре экрана. По вертикали его верхняя рамка соответствует переданному номеру строки экрана. При передаче недопустимой координаты поведение функции не определено.

В функции fexists_mes() используется интересный прием, иллюстрирующий примечательные возможности встроенного ассемблера. Суть его заключается в следующем. В составе функции производятся неоднократные вызовы функции make_hbar() для восстановления бруска курсора в окне предупреждения, причем каждый раз этой функции передаются одни и те же параметры. Для оптимизации работы программы можно было бы оформить повторяющиеся вызовы в виде отдельной функции без параметров. Но тогда пришлось бы все необходимые для этого переменные описывать на внешнем уровне, что при работе с большими моделями памяти (Compact, Large, Huge) и, следовательно, с far-адресами, существенно увеличило бы объем генерируемого кода и замедлило скорость выполнения. Даже при работе с моделью памяти Medium, где far-указатели используются только для функций, описывать повторяющиеся вызовы в виде отдельной функции представляется не слишком выгодным.

Избежать всех этих проблем позволяет встроенный ассемблер. Прямо в теле той функции, где производятся повторяющиеся вызовы, вы можете организовать процедуру встроенного ассемблера, которая и будет осуществлять необходимый вызов. Причем этот вызов вы можете производить при помощи кода Си и пользоваться теми же самыми локальными переменными, что и в основной функции, поскольку ассемблерная процедура не изменяет значение регистра BP. Так как ассемблерная процедура находится непосредственно в теле основной функции и, следовательно, в том же кодовом сегменте, то для ее вызова можно всегда использовать near-указатель. Таким образом, при работе с любой моделью памяти у вас всегда получится такой же результат, какой бы получился при организации для повторяющихся вызовов отдельной функции в моделях памяти Tiny и Small. Необходимо только отметить, что подобная ассемблерная процедура должна располагаться в таком месте, которого никогда не достигает выполняемый Си-код. Кроме этого нужно обязательно проследить за тем, чтобы у основной функции было зарезервировано в стеке место под локальные переменные (т.е. было произведено вычитание из указателя стека SP совокупной длины локальных переменных; этим обычно занимается компилятор).

Безусловно, у этого способа есть свои недостатки, связанные, например, с потерей оптимизации при использовании встроенного ассемблера. Но не вызывает сомнений, что программист, грамотно использующий все преимущества смешанного Си и ассемблерного кода, всегда сможет оптимизировать свою программу лучше, чем это сделает любой компилятор. Хотелось бы только предостеречь начинающих программистов о том, что такие трюки нужно делать крайне осторожно.

При компиляции функции fexists_mes() компилятор Turbo C++ 1.0 выдает массу предупреждений о том, что в функции использованы неопознанные ассемблерные инструкции, что обнаружен код, который никогда не достигается в ходе выполнения функции и т.п. Все эти предупреждения направлены на то, чтобы обратить внимание программиста на те места в программе, которые кажутся компилятору подозрительными. Если программист уверен в своих действиях, то такие предупреждения он может проигнорировать. Однако, если вы еще только начинающий программист (да и не только начинающий), никогда не следует отключать у компилятора опцию выдачи таких предупреждений.

```
/* Файл MAKEPRT.H */
/* Автор А.Синев, Copyright (C) 1990,1991 */
/* Turbo C 2.0, Turbo C + + 1.0 */

/* маска для выделения кода специальной клавиши,
   возвращаемого функцией get_choice() */
#define SpecialKeyMask 0x01FF

/* значение, возвращаемое функцией
   break_control(), при преждевременном выходе
   из программы */
#define ABORT 0
/* параметры для функции
   toggle_intensity_blinking() */
#define BLINKING 1
#define INTENSITY 0

#define ESCAPE 27      /* коды клавиш, */
#define CTRL_O 15      /* возвращаемые функцией */
#define BACKSPACE 8    /* getkey() */
#define ENTER 13
#define HOMEKEY 327
#define ENDKEY 335
#define UPKEY 328
#define DOWNKEY 336
#define PGUPKEY 329
#define PGDNKEY 337
#define LEFTKEY 331
#define RIGHTKEY 333
#define DELKEY 339

#define NoCursor 0x2000    /* отмена курсора */

int get_video_mode(void);
int get_cursor_size(void);
void set_cursor_size(int cursorshape);
void set_cursor_position(int column, int row);
unsigned long get_cursor_position_size(void);
void set_cursor_position_size(unsigned long);
void toggle_intensity_blinking(int sw);
int getkey(void);
int string_copy(char far *deststring,
    char far *sourcestring);
void make_hbar(int row,int startcol,int width,
     unsigned char far *sourcestring,
     unsigned char far *deststring);
void clear_nchars(int row,int startcol,
    int nn_chars);
void put_string(int row, int startcol,
```

```
    char far *sourcestring);
void update_right(int row,int startcol,
    int endcol,int begstatus,int endstatus,
    int cursorpos_w,char far *cursorpos_s);
void update_left(int row,int startcol,int endcol,
    int begstatus,int endstatus,int cursorpos_w,
    char far *cursorpos_s);
void insert_char(char far *sourcestring,
    int stringlength, int ch);
void delete_char(char far *sourcestring,
    int stringlength);
void make_window(int left,int top,int right,
    int bottom,char far *sourcetext,
    char far *buffer,int windowattr,
    int shadowbackgroundattr,int nn_hotkeys,
    int far *hotkeys, int hotkeyattr);
void get_window_text(int left,int top,int right.
    int bottom,char far *deststring);
void restore_text(int left,int top,int right,
            int bottom,char far *sourcetext);
int get_choice(int firstrow,int lastrow,
    int startcol,int barwidth,int currchoice,
    unsigned char *sourceattr,
    unsigned char *destattr,int nn_altkeys,
    int *altkeys,int bar_status);
int edit_string(int row,int startcol,int endcol,
    int cursorshape,int buffersize,
    char *originalstring,
    unsigned char *sourceattr,
    unsigned char *destattr);

int fexists_mes(int start_row,char *pathname);
void wildcard_mes(int start_row);

unsigned long break_off(void);
void break_on(unsigned long oldbreakvector);
int break_control(void);

void get_pathname(char *pattern,char *ff_name,
    char *pathname);
void shrink_fname(char *shrunkname,char *pathname,
    const char *default_extension);

void menu_on(void);
void menu_off(void);
void dialbox_on(void);
void dialbox_off(void);
int edit_fname(char *fname);
void edit_number(int *number);
void toggle_switch(int *sw);
void process(void);

#if defined(MAIN)

/* строка заголовка страницы */
#define HeadString "File: %-45s Page %d of %d\n"

#define NN_LinesStr "\
\nThe file %s consists of %u lines."
#define NN_PagesStr "\
\nThat will be %d pages of %d lines each."

/* основание системы счисления */
#define Radix 10
/* атрибуты файла для поиска */
#define FF_Attrib (FA_ARCH | FA_RDONLY)

/* строка текста окна меню (45x10) */
#define MenuText "\
----------------- MENU --------
Read from file
Write to file
Lines per page        60
Odd left margins      10
Even left margins     4
Write page Numbers    On
Screen output         On
Start Processing
----------------------------------

/* строка текста диалогового окна (43x3) */
#define DialBoxText "\
--------------- File Name ---------------


------------------------------------------

#define MenuBoxLeft 18    /* координаты окна */
#define MenuBoxTop 7      /* меню на экране */
#define MenuBoxRight 62
#define MenuBoxBottom 16

#define DialBoxLeft 19   /* координаты */
#define DialBoxRight 61  /* диалогового окна */

/* ширина курсора главного меню */
#define MenuBarWidth (MenuBoxRight-MenuBoxLeft-1)

/* размер буфера редактируемой строки */
#define StrBuffSize 65
/* левая координата обновляемой строки в окне
   меню и ее максимальная длина */
#define StrLeftCol 42
#define StrLength 19
/* размер буфера для редактирования чисел */
#define NumbBuffSize 4

/* строки индикации переключателей */
#define OnString "On "
#define OffString "Off"

/* модели имен файлов и их расширения */
#define InFNamePattern "*.TXT"
#define OutFNamePattern "*.PRT"
#define InFExtPattern ".TXT"
#define OutFExtPattern ".PRT"

/* массив кодов альтернативных клавиш */
#define NN_AltKeys 18
int AltKeys[NN_AltKeys] =
    { 'r','R','w','W','l','L','o','O','e','E',
      'n','N','s','S','p','P',ESCAPE,CTRL_O };

/* массив порядковых номеров символов строки
   текста меню (считая от единицы), которые
   в окне будут выделены другим цветом
   (атрибутами "горячих" клавиш) */
#define NN_HotChars 12
int HotCharNumbers[NN_HotChars] =
    {22,23,24,25,48,93,138,183,228,284,318,369};

/* буфер для сохранения участка экрана под
   окном меню (текст с атрибутами) */
char MenuBuffer[2*((MenuBoxRight-MenuBoxLeft + 1) +
        2)*((MenuBoxBottom-MenuBoxTop + 1) + 1)];
 /* массив атрибутов курсора главного меню */
unsigned char MenuBar[MenuBarWidth];
/* буфер для сохранения атрибутов экрана
   под курсором */
```

```
unsigned char MenuBarBuffer[MenuBarWidth],
/* буфер для сохранения участка экрана под
   диалоговым окном */
char DialBoxBuffer[2*
     ((DialBoxRight-DialBoxLeft + 1) + 2)*(3 + 1)];

/* массив атрибутов закрашиваемого бруска в
   диалоговом окне и буфер для старых атрибутов */
unsigned char DialBar[DialBoxRight-DialBoxLeft-2];
unsigned char DialBarBuffer[DialBoxRight-
                           DialBoxLeft-2];

/* массив атрибутов закрашиваемого бруска при
   редактировании чисел */
unsigned char NumbBar[NumbBuffSize-1];
unsigned char NumbBarBuff[NumbBuffSize-1];

/* значения атрибутов для окна главного меню,
   альтернативных (горячих) символов, диалогового
   окна, курсора в главном окне, тени главного
   окна, закрашиваемого бруска при редактировании
   чисел */
int MenuBoxAttr = BLACK + (LIGHTGRAY««4);
int HotAttr = RED + (LIGHTGRAY««4);
int DialBoxAttr = WHITE + (CYAN««4);
int BarAttr = WHITE + (BLACK««4);
int ShadowAttr = DARKGRAY««4;
int NumbBarAttr = YELLOW + (MAGENTA««4);

/* имена входного и выходного файлов */
char InFileName[StrBuffSize + 4];
char OutFileName[StrBuffSize + 4];

/* режимы открытия файла */
char WriteMode[] = "wt", AppendMode[] = "at";
char *OpenMode;

/* значения по умолчанию: число строк на
   странице, поля для нечетных и четных страниц,
   начальные состояния переключателей */
int LinesPerPage = 60;
int OddMargin = 10;
int EvenMargin = 4;
int Screen_sw = 1;
int PgNumb_sw = 1;

/* значение старого вектора прерывания 0x1B
   (Ctrl-Break) */
unsigned long OldBreakVector;

/* значение выбора в главном меню */
int Choice;

int VideoMode;   /* текущий видеорежим */
/* форма курсора: начальная скан-линия в старшем
   байте, конечная - в младшем */
int CursorShape;

/* указатель на структуру, используемую для
   сохранения и последующего восстановления
   значений регистров центрального процессора
   функциями setjmp() и longjmp() */
jmp_buf Jumper;

#endif

/* Конец файла MAKEPRT.H */
```

```
/* Файл MAIN.C */
/* Автор А.Синев, Copyright (C) 1990,1991 */
/* Turbo C 2.0, Turbo C++ 1.0 */
#define MAIN

#include «stdio.h»
#include «stdlib.h»
#include «conio.h»
#include «dir.h»
#include «dos.h»
#include «string.h»
#include «setjmp.h»

#include "makeprt.h"

/**********************************************/
void main(void)
{
 /* состояние курсора в окне: 0 - курсора нет */
 int bar_status = 0;
 /* коды дополнительных функциональных клавиш,
    отсутствующих в окне меню */
 int specialkey;
 /* полные имена файлов */
 char pathname[StrBuffSize + 4], *pthname;
 char shrunkname[20]; /* сокращенное имя файла */
 /* аргумент функции findfirst() */
 struct ffblk ffblk;
 /* значения, возвращаемые функциями
    setjmp() и longjmp() */
 int value;
 /* промежуточная переменная */
 int clrpattern = 0;

 /* установить альтернативную программу обработки
    прерывания 0x23 (Ctrl-Break handler) */
 ctrlbrk(break_control);

 /* временно отключить прерывание 0x1B
    (Ctrl-Break) */
 OldBreakVector = break_off();

 /* прочитать текущий режим дисплея и форму
    курсора */
 VideoMode = get_video_mode();
 CursorShape = get_cursor_size();

 /* если дисплей в монохромном режиме - поменять
    значения атрибутов, иначе переключить бит
    мигания/интенсивности */
 if (VideoMode == MONO) {
   MenuBoxAttr = DialBoxAttr =
   NumbBarAttr = LIGHTGRAY + (BLACK««4);
   HotAttr = LIGHTBLUE + (BLACK««4);
   BarAttr = BLACK + (LIGHTGRAY««4);
   ShadowAttr = BLACK««4;
 } else
   toggle_intensity_blinking(INTENSITY);

 /* временно спрятать курсор */
 set_cursor_size(NoCursor);

 /* заполнить атрибутами массивы курсоров */
 memset(MenuBar,BarAttr,MenuBarWidth);
 memset(DialBar,BarAttr,
     DialBoxRight-DialBoxLeft-2);
 memset(NumbBar,NumbBarAttr,NumbBuffSize-1);

 /* начальный режим открытия выходного файла */
```


4 Зак. 2125

```
OpenMode = WriteMode;

Choice = 1;      /* начальное значение выбора */
menu_on();       /* включить окно меню */

while (1) {      /* бесконечный цикл */
 /* получить значение выбора и производить
    соответствующую обработку */
 switch (Choice = get_choice(MenuBoxTop + 1,
    MenuBoxBottom-1,MenuBoxLeft + 1,MenuBarWidth,
    Choice,MenuBar,MenuBarBuffer,NN_AltKeys,
    AltKeys,bar_status)) {
  /* редактировать имя входного файла */
  case 1:
   bar_status = 1;      /* курсор меню есть */
   /* построить диалоговое окно */
   dialbox_on();
   /* скопировать модель имени файла */
   if (! *InFileName) {
     strcpy(InFileName,InFNamePattern);
     clrpattern = 1;
   }
   /* редактировать имя входного файла */
   if (!edit_fname(InFileName)) {
     /* восстановить экран под диалоговым
        окном */
     dialbox_off();
     if (clrpattern)
       *InFileName = 0;
     clrpattern = 0;
     break;
   }
   clrpattern = 0;
   /* сократить имя входного файла для выдачи
      на экран */
   if (*InFileName)
     shrink_fname(shrunkname,InFileName,
                        InFExtPattern);
   else
     *shrunkname = 0;
   /* восстановить экран под диалоговым окном */
   dialbox_off();
   /* очистить место в окне для имени файла */
   clear_nchars(MenuBoxTop + Choice,StrLeftCol,
                        StrLength);
   /* выдать на экран имя входного файла */
   put_string(MenuBoxTop + Choice,StrLeftCol,
                        shrunkname);
   break;
  /* редактировать имя выходного файла */
  case 2:
   bar_status = 1;
   /* скопировать модель имени файла */
   if (! *OutFileName)
     strcpy(OutFileName,OutFNamePattern);
   /* построить диалоговое окно */
   dialbox_on();
   /* редактировать имя выходного файла */
   if (!edit_fname(OutFileName)) {
     /* восстановить экран под окном */
     dialbox_off();
     break;
   }
   /* сократить имя выходного файла для выдачи
      на экран */
   if (*OutFileName)
     shrink_fname(shrunkname,OutFileName,
                        OutFExtPattern);
   else
     *shrunkname = 0;
   /* если в имени файла есть символы *,?
      выдать предупреждение */
   if (strchr(OutFileName,'*') != NULL ||
         strchr(OutFileName,'?') != NULL) {
     wildcard_mes(MenuBoxTop + Choice + 1);
     *OutFileName = *shrunkname = 0;
   }
   /* если файл с указанным именем уже имеется
      на диске */
   if (!findfirst(OutFileName,&ffblk,
                        FF_Attrib)) {
     /* получить полное имя файла */
     get_pathname(OutFileName,ffblk.ff_name,
                        pathname);
     pthname = searchpath(pathname);
     /* выдать предупреждение и получить
        режим открытия файла */
     switch (fexists_mes(MenuBoxTop + Choice + 1,
                        pthname)) {
      case 0: OpenMode = WriteMode;
        break;
      case 1: OpenMode = AppendMode;
        break;
      default: OpenMode = WriteMode;
        *OutFileName = *shrunkname = 0;
     }
   }
   /* восстановить экран под окном */
   dialbox_off();
   /* очистить место в окне для имени файла */
   clear_nchars(MenuBoxTop + Choice,StrLeftCol,
                        StrLength);
   /* выдать на экран имя выходного файла */
   put_string(MenuBoxTop + Choice,StrLeftCol,
                        shrunkname);
   break;
  /* редактировать число строк на страницу
     выходного файла */
  case 3:
   bar_status = 1;
   edit_number(&LinesPerPage);
   break;
  /* редактировать ширину полей для нечетных
     страниц */
  case 4:
   bar_status = 1;
   edit_number(&OddMargin);
   break;
  /* редактировать ширину полей для четных
     страниц */
  case 5:
   bar_status = 1;
   edit_number(&EvenMargin);
   break;
  /* переключить ключ выдачи номеров страниц */
  case 6:
   bar_status = 1;
   toggle_switch(&PgNumb_sw);
   break;
  /* переключить ключ выдачи текста на экран */
  case 7:
   bar_status = 1;
   toggle_switch(&Screen_sw);
   break;
  /* обрабатывать текст */
  case 8:
   menu_off();        /* погасить окно меню */
   /* восстановить курсор */
```

```
    set_cursor_size(CursorShape);
    /* восстановить вектор прерывания 0x1B */
    break_on(OldBreakVector);
    /* установить адрес перехода для возврата
       из функции обработки прерывания 0x23
       (Ctrl-Break handler) */
    value = setjmp(Jumper);
    if (!value)
      process();        /* обрабатывать текст */
    /* отключить обработку прерывания 0x1B */
    OldBreakVector = break_off();
    /* отменить курсор */
    set_cursor_size(NoCursor);
    menu_on();          /* включить окно меню */
    bar_status = 0;     /* курсора меню нет */
    break;
  /* обработка дополнительных функциональных
     клавиш, отсутствующих в окне меню */
  default:
    /* получить код дополнительной клавиши */
    specialkey = Choice & SpecialKeyMask;
    /* получить значение текущего выбора */
    Choice >>= 9;
    switch (specialkey) {
     case ESCAPE:   /* нажата клавиша ESCAPE */
       menu_off();      /* погасить окно меню */
       /* восстановить бит
          мигания/интенсивности */
       if (VideoMode != MONO)
         toggle_intensity_blinking(BLINKING);
       /* восстановить курсор */
       set_cursor_size(CursorShape);
       /* восстановить вектор прерывания
          0x1B (Ctrl-Break) */
       break_on(OldBreakVector);
       return;      /* выход из программы */
     /* нажата клавиша Ctrl-O */
     case CTRL_O:
       menu_off();  /* погасить окно меню */
       getkey();    /* ждать нажатия клавиши */
       menu_on();   /* включить окно меню */
       bar_status = 0;  /* курсора меню нет */
     }
  }
 }
} /* main */

/******** Функция включения окна меню *********/
void menu_on(void)
{
 /* построить окно на экране */
 make_window(MenuBoxLeft,MenuBoxTop,
      MenuBoxRight,MenuBoxBottom,MenuText,
      MenuBuffer,MenuBoxAttr,ShadowAttr,
      NN_HotChars,HotCharNumbers,HotAttr);
} /* menu_on() */

/******** Функция выключения окна меню ********/
void menu_off(void)
{
 /* сохранить текст окна меню (без атрибутов) */
 get_window_text(MenuBoxLeft,MenuBoxTop,
        MenuBoxRight,MenuBoxBottom,MenuText);
 /* восстановить экран из буфера */
 restore_text(MenuBoxLeft,MenuBoxTop,
     MenuBoxRight + 2,MenuBoxBottom + 1,MenuBuffer);
} /* menu_off() */

/***** Функция включения диалогового окна *****/
void dialbox_on(void)
{
 /* построить окно */
 make_window(DialBoxLeft,(MenuBoxTop + Choice + 1),
          DialBoxRight,(MenuBoxTop + Choice + 3),
          DialBoxText,DialBoxBuffer,
          DialBoxAttr,0,0,0,0);
} /* dialbox_on() */

/***** Функция выключения диалогового окна ****/
void dialbox_off(void)
{
 /* восстановить экран из буфера */
 restore_text(DialBoxLeft,(MenuBoxTop + Choice + 1),
        DialBoxRight + 2,(MenuBoxTop + Choice + 3) + 1,
                         DialBoxBuffer);
} /* dialbox_off() */

/***** Функция редактирования имени файла *****/
int edit_fname(char *fname)
{
 /* редактировать строку */
 return (edit_string((MenuBoxTop + Choice + 1) + 1,
     DialBoxLeft + 1,DialBoxRight-1,CursorShape,
     StrBuffSize,fname,DialBar,DialBarBuffer));
} /* edit_fname() */

/***** Функция редактирования целого числа ****/
void edit_number(int *number)
{
 char buffer[NumbBuffSize];   /* буфер строки */
 int numberlength;            /* длина строки */

 /* преобразовать число в строку и вычислить
    ее длину */
 itoa(*number,buffer,Radix);
 numberlength = strlen(buffer);
 /* очистить место в окне под строку */
 clear_nchars(MenuBoxTop + Choice,StrLeftCol,
                          NumbBuffSize);
 /* редактировать строку */
 edit_string(MenuBoxTop + Choice,StrLeftCol-1,
     StrLeftCol + 4,CursorShape,NumbBuffSize,
     buffer,NumbBar,NumbBarBuff);

 /* число не может быть отрицательным */
 if ((*number = atoi(buffer)) < 0)
   *number = 0;
 /* снова преобразовать целое в строку; это
    необходимо для исключения ошибки ввода
    неправильной строки */
 itoa(*number,buffer,Radix);
 /* очистить место в окне для строки */
 clear_nchars(MenuBoxTop + Choice,StrLeftCol,
                          NumbBuffSize);
 /* вывести строку на экран */
 put_string(MenuBoxTop + Choice,StrLeftCol,buffer);
 /* восстановить атрибуты строки на экране */
 make_hbar(MenuBoxTop + Choice,StrLeftCol,
        numberlength,NumbBarBuff,
        (unsigned char *)buffer);
} /* edit_number() */

/********* Функция переключения опции *********/
void toggle_switch(int *sw)
{
 /* указатель на строку индикации */
 char *swstrptr;
```

```
 if (*sw) {
   *sw = 0;
   swstrptr = OffString;
 } else {
   *sw = 1;
   swstrptr = OnString;
 }
 /* вывести строку индикации на экран */
 put_string(MenuBoxTop + Choice,StrLeftCol,
                         swstrptr);
} /* toggle_switch() */

/***** Функция обработки прерывания 0x23 ******/
/*********** (Ctrl-Break handler) *************/
int break_control(void)
{
 fputs("User Break ...\n",stdout);
 /* закрыть все открытые потоки */
 if ((fcloseall()) != EOF)
   /* перейти на адрес возврата, установленый
      в структуре *Jumper функцией setjmp () */
   longjmp(Jumper,1);
 /* если ошибка, то выход из программы */
 perror("\nError");
 /* восстановить бит мигания/интенсивности */
 if (VideoMode != MONO)
   toggle_intensity_blinking(BLINKING);
 return ABORT;        /* выход из программы */
} /* break_control() */

/****** Функция разбивки текстовых файлов *****/
/****** на страницы с добавлением полей *******/
void process(void)
{
 FILE *infile, *outfile;
 int ofstat = 0; /* состояние выходного файла */
 /* количество строк во входном файле */
 unsigned int nn_lines;
 int nn_pages;          /* количество страниц */
 /* указатель на структуру описания файла */
 struct ffblk ffblk;
 /* прочитанный символ, счетчики, поля */
 int ch, i, j, k, margin;
 /* имена найденных файлов */
 char *pathname, inff_name[StrBuffSize + 4];

 /* открыть выходной поток */
 if (*OutFileName) {
   if ((outfile = fopen(OutFileName,OpenMode)) !=
                                     NULL)
     ofstat = 1;
   else {
     printf("\nCan't open output file %s",
                         OutFileName);
     perror("");
   }
 }
 if (!ofstat)
   fputs("\nNo output file!",stdout);

 /* искать входной файл, соответствующий заданной
    модели имени */
 if (!findfirst(InFileName,&ffblk,FF_Attrib)) {
   /* если выходной файл открыт в режиме "at",
      то начать новую страницу */
   if (ofstat && *OpenMode == 'a')
     fputs("\n\f\n",outfile);

/* метка перехода, если найден следующий файл,
   соответствующий заданной модели */
next_file:
   /* получить полное имя найденного файла */
   get_pathname(InFileName,ffblk.ff_name,
                          inff_name);
   pathname = searchpath(inff_name);

   /* открыть входной поток */
   if ((infile = fopen(pathname,"rt")) != NULL) {
     printf("\n\nReading from file: %s ...",
                              pathname);
     nn_lines = nn_pages = 0; /* нач. значения */
     /* если установлен счетчик страниц */
     if (PgNumb_sw) {
       fputs("\nAnalysing ...",stdout);
       do {
         if ((ch = fgetc(infile)) == '\n')
           nn_lines++; /* считать число строк */
       } while (ch != EOF);
       nn_pages = LinesPerPage ?
         (nn_lines + 1) / LinesPerPage + 1 : 1;
       printf(NN_LinesStr,pathname,nn_lines + 1);
       printf(NN_PagesStr,nn_pages,LinesPerPage);
       rewind(infile);
     } else
       nn_pages = 0x7FFF;

     /* перевести строку, если разрешен вывод на
        экран */
     if (Screen_sw)
       fputs("\n\n",stdout);

     for (i = 0; i<nn_pages; i++) {
       /* определить размер полей для текущей
          страницы */
       margin = i & 0x0001 ?
             EvenMargin : OddMargin;
       /* если не первая страница, то вывести
          символ перевода формата */
       if (i) {
         if (ofstat)
           fputs("\f\n",outfile);
         if (Screen_sw)
           fputs("\f\n",stdout);
       }
       /* вывести номер страницы */
       if (PgNumb_sw) {
         if (ofstat) {
           fprintf(outfile,"%*s",margin,"");
           fprintf(outfile,HeadString,pathname,
                        i + 1,nn_pages);
           fprintf(outfile,"%*s",margin,"");
           for (k = 0; k<65; k++)
             fputc('-',outfile);
           fputc('\n',outfile);
         }
         if (Screen_sw) {
           fprintf(stdout,"%*s",margin,"");
           fprintf(stdout,HeadString,pathname,
                        i + 1,nn_pages);
           fprintf(stdout,"%*s",margin,"");
           for (k = 0; k<65; k++)
             fputc('-',stdout);
           fputc('\n',stdout);
         }
       }
       /* выводить текст */
       for (j = 0; j<LinesPerPage; j++) {
         /* вставить поля */
```

```
        if (ofstat)
          fprintf(outfile,"%*s",margin,"");
        if (Screen_sw)
          fprintf(stdout,"%*s",margin,"");
        /* выводить строку */
        do {
          /* если конец файла, то закрыть
             текущий входной поток и искать
             следующий файл, соответствующий
             заданной модели имени */
          if ((ch = fgetc(infile)) == EOF) {
            fclose(infile);
            if (!findnext(&ffblk)) {
              /* если файл найден, то
                 перевести формат и вернуться
                 в начало цикла */
              if (ofstat)
                fputs("\n\f\n",outfile);
              if (Screen_sw)
                fputs("\n\f\n",stdout);
              goto next_file;
            } else       /* иначе возврат */
              goto quit;
          }
          /* вывести прочитанный символ */
          if (ofstat)
            fputc(ch,outfile);
          if (Screen_sw)
            fputc(ch,stdout);
        } while (ch != '\n');
      }
    }
  } else {
    /* если ошибка открытия входного потока */
    printf("\nCan't open input file %s",
                                pathname);
    perror("");
  }
 } else        /* если входной файл не найден */
   printf("\nThe file %s not found.",InFileName);
quit:
 if (ofstat)   /* закрыть выходной поток */
   fclose(outfile);
} /* process() */

/* Конец файла MAIN.C */

/* Файл UTILIT1.C */
/* Автор А.Синев, Copyright (C) 1990,1991 */
/* Turbo C 2.0, Turbo C++ 1.0 */

#include <string.h>
#include <conio.h>
#include <alloc.h>

#include "makeprt.h"

/********** Получить полное имя файла **********/
/* Функция заполняет буфер *pathname строкой
   полного имени файла, составляемой из пути
   поиска (определяемого по заданной модели
   *pattern) и имени файла *ff_name (найденного,
   например, при помощи функции findnext()).
   Размер буфера должен быть достаточным для
   размещения в нем возвращаемой строки. */

void get_pathname(char *pattern,char *ff_name,
                            char *pathname)
{
 /* указатель на последний символ '\\' в строке
    модели поиска */
 char *last_bslash;

 /* скопировать модель поиска в выходную строку */
 strcpy(pathname,pattern);

 if ((last_bslash = strrchr(pathname,'\\')) != NULL)
   last_bslash[1] = 0;
 else
   *pathname = 0;
 /* присоединить имя файла к пути поиска */
 strcat(pathname,ff_name);
} /* get_pathname() */

/********* Функция сжатия имени файла *********/
/* Функция сжимает имя файла до максимальной
   длины 19 байт + 1 байт (0), добавляя при этом
   необходимое расширение *def_ext (если таковое
   отсутствовало в переданном имени), и заполняет
   строку *shrunkname полученным текстом. В итоге
   строка приобретает вид C:\...\FILENAME.EXT
   (в зависимости от переданного имени).
   Параметры:
   *shrunkname - указатель на буфер для заполнения
              (размер буфера - 20 байт)
   *pathname - указатель на первоначальное имя
            файла
   *def_ext - указатель на строку присоединяемого
           расширения (вида ".EXT"), либо на
           нулевую строку */

void shrink_fname(char *shrunkname,
          char *pathname,const char *def_ext)
{
 /* указатели на первый и последний символ '\\'
    в переданной строке */
 char *firstbs,*lastbs;
 /* указатель на последний символ ':' */
 char *lastcolon;
 /* указатель на первый символ '.'
    непосредственно в имени файла */
 char *firstpoint;
 /* указатель на текущий символ в переданной
    строке */
 char *charptr;
 /* текущий символ в заполняемой строке */
 int currentchar;
 /* длины имени файла и расширения */
 char name_length, ext_length;
 register int i;  /* счетчик */

 charptr = pathname;
 currentchar = 0;

 if ((lastcolon = strrchr(charptr,':')) != NULL) {
   if (lastcolon > pathname) {
     shrunkname[0] = lastcolon[-1];
     shrunkname[1] = lastcolon[0];
     currentchar = 2;
     charptr = lastcolon + 1;
   }
 }
 if ((lastbs = strrchr(charptr,'\\')) != NULL) {
   firstbs = strchr(charptr,'\\');
   if (lastbs != firstbs)
     shrunkname[currentchar++] = '\\';
   if (lastbs != charptr) {
     shrunkname[currentchar++] = '.';
```

```
    shrunkname[currentchar++] = '.';
    shrunkname[currentchar++] = '.';
  }
  shrunkname[currentchar++] = '\\';
  charptr = lastbs + 1;
}
if ((firstpoint = strchr(charptr,'.')) != NULL) {
  name_length = firstpoint - charptr;
  if (name_length > 8)
     name_length = 8;
  for (i = 0; i<name_length;
      shrunkname[currentchar++] = charptr[i++]);
  shrunkname[currentchar++] = '.';
  charptr = firstpoint + 1;
  if ((ext_length = strlen(charptr)) > 3)
    ext_length = 3;
  for (i = 0; i<ext_length;
      shrunkname[currentchar++] = charptr[i++]);
  shrunkname[currentchar] = 0;
} else {
  if ((name_length = strlen(charptr)) > 8)
    name_length = 8;
  for (i = 0; i<name_length;
      shrunkname[currentchar++] = charptr[i++]);
  shrunkname[currentchar] = 0;
  strcat(pathname,def_ext);
  strcat(shrunkname,def_ext);
}
} /* shrink_fname() */


#define WildcardLeft 26    /* левая и правая */
#define WildcardRight 54   /* координаты окна */
/* строка текста предупреждения (29x5) */
#define WildcardText "\
---------------------------
 You can't use wildcards
 in the output file name.
       Press ESC.
---------------------------

/**** Функция выдачи окна с предупреждением ****/
/* о наличии символов *,? в модели имени файла */
/* start_row - номер строки экрана,
        соответствующей верхней рамке окна */

void wildcard_mes(int start_row)
{
 char *wildcardbuff;      /* буфер экрана */
 int bottom_row;    /* нижняя координата окна */
 char attr = WHITE + (RED<<4); /*атрибуты окна*/

 /* зарезервировать буфер */
 if ((wildcardbuff = malloc((29+2)*(5+1)*2)) ==
                                     NULL)
   return;
 /* если монохромный режим, то сменить атрибуты */
 if (get_video_mode() == MONO)
   attr = LIGHTGRAY;
 /* вычислить нижнюю координату окна */
 bottom_row = start_row + 4;
 /* построить окно */
 make_window(WildcardLeft,
     start_row,WildcardRight,bottom_row,
     WildcardText,wildcardbuff,attr,0,0,0,0);
 /* ждать нажатия клавиши ESCAPE */
 while (getkey() != ESCAPE);
 /* восстановить экран под окном */
 restore_text(WildcardLeft,start_row,
     WildcardRight+2,bottom_row+1,wildcardbuff):
} /* wildcard_mes() */

/* Конец файла UTILIT1.C */

/* Файл UTILIT2.C */
/* Автор А.Синев, Copyright (C) 1990,1991 */
/* Turbo C 2.0, Turbo C++ 1.0 */

#pragma inline

#include <string.h>
#include <conio.h>
#include <alloc.h>

#include "makeprt.h"

#define LeftUpper  'т'
#define HorizBar   '-'
#define RightUpper 'я'
#define LeftLower  'м'
#define VertBar    'у'
#define RightLower 'с'
#define TheFile_Str "The file "
#define TheFile_StrLen 9
#define AlreadyExists_Str "already exists."
#define AlreadyExists_StrLen 15
#define Choice_Str " Overwrite  Append  Cancel "
#define Choice_StrLen 29
#define MaxLength  41
#define MaxPathNameLength 32
#define FirstChoiceWidth 11
#define SecondChoiceWidth 8
#define ThirdChoiceWidth 8
#define SecondChoiceOffset 12
   #define ThirdChoiceOffset 21
#define ChoiceWidths "\x0b\x08\x08"  /* 11,8,8 */

/* у The file C:\...\12345678.123\12345678.123 у
      123456789012345678901234567890123456789012345
  Максимальный размер окна = 45 x 5 (без тени)
*/
/****** Функция сообщения о существовании *****/
/********** указанного файла на диске *********/
/* Функция строит окно, горизонтальные размеры
  которого настраиваются под длину переданного
  имени файла и возвращает выбранный режим
  открытия файла:
  0 - Overwrite (перезаписать),
  1 - Append (добавить к концу),
  2 - Cancel (отмена).
  Параметры:
  start_row - номер строки экрана,соответствую-
          щей верхней рамке окна сообщения;
  *pathname - указатель на имя файла.
*/
int fexists_mes(int start_row, char *pathname)
{
 /* буферы текста и экрана */
 char *window_text, *window_buff;
 /* буфер сокращенного имени файла */
 char shrunkname[MaxPathNameLength];
 /* буферы курсора */
 unsigned char bar[FirstChoiceWidth];
 unsigned char bar_buff[FirstChoiceWidth];
 /* указатели на имя файла и текущий символ
   имени */
 char *fnptr, *currptr;
 /* длина переданного имени файла */
```

```
int pnlength;
int fnlength;    /* длина имени файла в окне */
int currentchar; /* счетчик символов */
/* ширина и координаты окна */
int windowwidth, windowleft;
int windowright, windowbottom;
/* промежуточные переменные */
int firststrlen, textlen;
int leftmargin, rightmargin;
int choice, choicerow, choiceleft[3];
int hotkeys[3];
/* атрибуты окна */
char attr = WHITE + (RED««4);
/* атрибуты "горячих" клавиш */
char hotattr = YELLOW + (RED««4);
register int i,j;             /* счетчики */

/* зарезервировать буферы под текст окна и
   участок экрана под окном */
if ((window_text = malloc(45*5)) = = NULL)
  return 2;
if ((window_buff = malloc((45 + 2)*(5 + 1)*2)) = =
                                       NULL) {
  free(window_text);
  return 2;
}
/* если монохромный режим, то сменить атрибуты */
if (get_video_mode() = = MONO) {
  attr = LIGHTGRAY;
  hotattr = LIGHTBLUE;
}

/* Сократить имя файла под максимальную ширину
   окна (если необходимо) */
if ((pnlength = strlen(pathname)) « =
                      MaxPathNameLength) {
  fnptr = pathname;
  fnlength = pnlength;
} else {
  currentchar = 0;
  currptr = pathname;
  if (strrchr(currptr,':') = = currptr + 1) {
    shrunkname[currentchar + + ] = currptr[0];
    shrunkname[currentchar + + ] = currptr[1];
    currptr + = 2;
  }
  if (strchr(currptr,'\\') = = currptr)
    shrunkname[currentchar + + ] = '\\';
  shrunkname[currentchar + + ] = '.';
  shrunkname[currentchar + + ] = '.';
  shrunkname[currentchar + + ] = '.';
  for (i = MaxPathNameLength-1, j = pnlength-1;
           i » = currentchar;
           shrunkname[i--] = pathname[j--]);
  fnptr = shrunkname;
  fnlength = MaxPathNameLength;
}
/* длина первой строки, записываемой в окно */
firststrlen = TheFile_StrLen + fnlength;
/* ширина текста в окне */
textlen = (firststrlen » Choice_StrLen) ?
         firststrlen + 2 : Choice_StrLen + 2;

/* заполнить атрибутами массив курсора */
memset(bar,LIGHTGRAY««4,FirstChoiceWidth);

/* заполнить первую строку окна */
i = 0;
window_text[i + + ] = LeftUpper;
memset(window_text + i,HorizBar,textlen);
i + = textlen;
window_text[i + + ] = RightUpper;

/* заполнить вторую строку окна */
leftmargin = (textlen - firststrlen) »» 1;
rightmargin = textlen - firststrlen - leftmargin;
window_text[i + + ] = VertBar;
memset(window_text + i,' ',leftmargin);
i + = leftmargin;
memcpy(window_text + i,TheFile_Str,TheFile_StrLen);
i + = TheFile_StrLen;
memcpy(window_text + i,fnptr,fnlength);
i + = fnlength;
memset(window_text + i,' ',rightmargin);
i + = rightmargin;
window_text[i + + ] = VertBar;

/* заполнить третью строку окна */
leftmargin = (textlen-AlreadyExists_StrLen)»»1;
rightmargin = textlen - AlreadyExists_StrLen -
                              leftmargin;
window_text[i + + ] = VertBar;
memset(window_text + i,' ',leftmargin);
i + = leftmargin;
memcpy(window_text + i,AlreadyExists_Str,
                    AlreadyExists_StrLen);
i + = AlreadyExists_StrLen;
memset(window_text + i,' ',rightmargin);
i + = rightmargin;
window_text[i + + ] = VertBar;

/* заполнить четвертую строку окна */
choiceleft[0] = (textlen - Choice_StrLen) »» 1;
rightmargin = textlen - Choice_StrLen -
                              choiceleft[0];
window_text[i + + ] = VertBar;
memset(window_text + i,' ',choiceleft[0]);
i + = choiceleft[0];
memcpy(window_text + i,Choice_Str,Choice_StrLen),
i + = Choice_StrLen;
memset(window_text + i,' ',rightmargin);
i + = rightmargin;
window_text[i + + ] = VertBar;

/* заполнить пятую строку окна */
window_text[i + + ] = LeftLower;
memset(window_text + i,HorizBar,textlen);
window_text[i + textlen] = RightLower;

/* вычислить координаты окна на экране */
windowwidth = textlen + 2;
windowleft = (80 - windowwidth) »» 1;
windowright = windowleft + windowwidth;
windowleft + + ;
windowbottom = start_row + 4;

/* вычислить номера "горячих" символов в строке
   текста окна */
hotkeys[0] = windowwidth*3 + choiceleft[0] + 3;
hotkeys[1] = hotkeys[0] + SecondChoiceOffset;
hotkeys[2] = hotkeys[0] + ThirdChoiceOffset;

/* вычислить координаты курсора */
choicerow = windowbottom - 1;
choiceleft[0] + = windowleft + 1;
choiceleft[1] = choiceleft[0] + SecondChoiceOffset;
choiceleft[2] = choiceleft[0] + ThirdChoiceOffset;
```

```
 /* построить окно */
 make_window(windowleft,start_row,windowright,
         windowbottom,window_text,window_buff,
         attr,0,3,hotkeys,hotattr);
 choice = 0;    /* начальное значение выбора */

loop:
 /* построить курсор */
 make_hbar(choicerow,choiceleft[choice],
         ChoiceWidths[choice],bar,bar_buff);
key_loop:
 /* прочитать код с клавиатуры */
 switch(getkey()) {
  case ENTER:
    break;
  case HOMEKEY:
    /* восстановить атрибуты под курсором (вызов
      процедуры встроенного ассемблера) */
    asm call near ptr restorecursor
    choice = 0;
    goto loop;      /* возврат в цикл */
  case LEFTKEY:
    /* восстановить атрибуты под курсором */
    asm call near ptr restorecursor
    choice--;
    if (choice < 0)
      choice = 2;
    goto loop;      /* возврат в цикл */
  case ENDKEY:
    asm call near ptr restorecursor
    choice = 2;
    goto loop;
  case RIGHTKEY:
    asm call near ptr restorecursor
    choice + + ;
    if (choice > 2)
      choice = 0;
    goto loop;
  case 'o':
  case 'O':
    choice = 0;
    break;
  case 'a':
  case 'A':
    choice = 1;
    break;
  case 'c':
  case 'C':
  case ESCAPE:
    choice = 2;
    break;
  default:              /* возврат в цикл */
    goto key_loop;      /* чтения с клавиатуры */
 }
 /* восстановить экран под окном */
 restore_text(windowleft,start_row,
     windowright + 2,windowbottom + 1,window_buff);
 free(window_text);    /* освободить буферы */
 free(window_buff);
 return choice;    /* вернуть значение выбора */

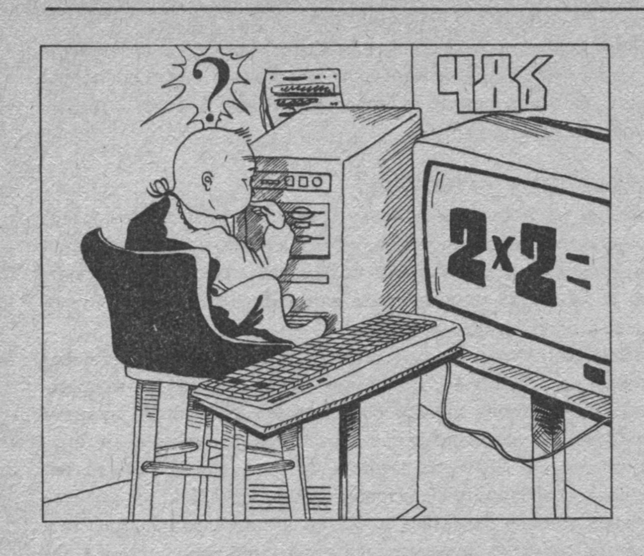
/*---------- Процедура встроенного ассемблера */
/* Восстановление атрибутов экрана под курсором */
 asm restorecursor proc near
 make_hbar(choicerow,choiceleft[choice],
         ChoiceWidths[choice],bar_buff,bar);
 asm  ret
 asm restorecursor endp

} /* fexists_mes() */

/* Конец файла UTILIT2.C */
```

А. Синев

(Окончание следует)



Стало уже традицией, что каждые 2-3 года Intel представляет на суд уже ничему не удивляющихся фирм-изготовителей компьютеров нового члена знаменитого семейства микропроцессоров 80X86. И каждое такое событие неизбежно порождает почти что гамлетовский вопрос, волнующий потенциальных покупателей, то есть нас с вами: менять или не менять?

Кому нужен этот i486

Время идет, первые 8086/8088 сменились более производительными i286, которые, в свою очередь, уступили место 32-разрядным кристаллам серии 80386, а сегодня уже и сверхмощные i486 перестали кого-либо удивлять. Так сравнимы ли преимущества новых систем с затратами на их приобретение?

Качественный скачок

Когда фирма Intel объявила о выпуске процессора 80286, это было воспринято как качественный скачок в микропроцессорной технологии. Предполагалось, что новый кристалл будет обладать такими особенностями, которых принципиально был лишен процессор 8088. К ним можно отнести и высокую тактовую частоту, и 16-разрядную шину данных, и режим работы с защитой, позволяющий реализовать многопользовательскую систему на персональном компьютере. Последнее, правда, удалось осуществить только в следующем поколении: i386 оказался действительно многозадачным и многопользовательским процессором, обладающим, к тому же, 32-разрядной шиной данных, существенно более высокой тактовой частотой и возможностью практически безграничной адресации.

Кристалл i486, в свою очередь, создавался как своего рода "продолжатель" линии 80386 (кстати, сама фирма Intel определяет процессор 80486 как супер-386). Тем не менее, объединив в одном корпусе почти 1,2 миллиона транзисторов, удалось создать микросхему, по своим параметрам перекрывающую возможности минимашины или даже универсального компьютера. В то же время i486 полностью совместим на программном уровне со своими "старшими братьями" — процессорами серии 80386.

Большинство усовершенствований, характерных для i486, носят чисто аппаратный характер. Давайте попытаемся разобраться, почему 25-мегагерцевый 80486 работает в 3-4 раза производительнее процессора 80386, рассчитанного на ту же тактовую частоту. Это связано с тем, что выполнение одной и той же инструкции в первом случае производится за 1-2 цикла, а во втором — за 4-5. К тому же в корпусе i486 помимо центрального процессора находятся математический сопроцессор, кэш и специальное устройство управления памятью, позволяющее физически адресовать до 4 Гбайт ОЗУ, при этом виртуальный адрес может достигать 64 Тбайт. Богатая система 32-разрядных команд, а также некоторые встроенные функции позволяют использовать процессор 80486 в мультипрограммных и многопользовательских приложениях.

Миникомпьютер в одной микросхеме

Фирма Intel опубликовала бюллетень, содержащий результаты измерений производительности нескольких вычислительных систем, построенных на базе 25- и 33-мегагерцевых процессоров i486. Сравнение некоторых характеристик показывает, что в ряде случаев персональный компьютер не уступает мини-ЭВМ DEC VAX 8550 размером с два холодильника и стоимостью под 100000 долларов.

К сожалению, подобное тестирование не отражает реальной производительности вычислительной системы, так как

5 Зак. 2125

проводится в строго определенных условиях и далеко не по всем показателям. Поэтому, хотя процессор i486 обладает впечатляющей вычислительной мощностью, существует множество факторов, влияющих на результирующую скорость работы компьютера.

Реальная производительность компьютера с таким быстрым процессором, как i486, ограничивается отнюдь не возможностями самого кристалла, а способностью вычислительной системы передавать информацию с соответствующей скоростью. Другими словами, ОЗУ, видеоадаптер, жесткий диск и сама шина являются как раз тем "болотом", в котором "вязнет" спринтер-процессор. Ситуация до боли напоминает езду на "Мерседесе" по колдобистым московским переулкам. Одним из способов разрешения подобного конфликта может быть использование специальных процессоров ввода/вывода, существенно сокращающих время вынужденного простоя центрального процессора. Очевидно, что простая замена i386 на i486 без соответствующей доработки остальных узлов вычислительной системы к повышению производительности не приведет.

Скорость доступа к ОЗУ в большинстве систем на базе i486 близка к 70 наносекундам, в то время как процессор может выполнять команды чтения/записи в несколько раз быстрее. Встроенная 8-килобайтная кэш-память играет роль буфера между относительно медленной основной памятью и скоростным кристаллом. Практически данные, предназначенные для записи в ОЗУ, сначала помещаются в кэш и, пока процессор выполняет следующую инструкцию, спокойно и неторопливо перекачиваются по указанному адресу. Кэширование операции чтения — вещь куда более хитрая: программа управления кэш-памятью должна предвидеть, какие именно данные должны быть считаны из ОЗУ за несколько циклов перед тем, как они понадобятся процессору. По мнению специалистов фирмы Intel, кэширование экономит до 80% времени при обращении к оперативной памяти.

Большинство производителей вычислительной техники считает, что встроенных 8 Кбайт кэш отнюдь не достаточно и, в связи с этим, комплектуют свои системы 25-наносекундной кэш-памятью объемом 128 Кбайт. С правомерностью подобных рассуждений можно согласиться, особенно если учесть тех пользователей, которым компьютер необходим для таких применений, как САПР или настольная типография. Но тут на первый план выступает производительность видеоадаптера, на который в этом случае ложится максимальная нагрузка.

Теперь о жестком диске. Большинство программных продуктов построены таким образом, что их функционирование требует интенсивного использования дискового пространства. Но стоит нам сравнить время доступа к жесткому диску с временем выполнения одной команды типа регистр-регистр, как становится очевидной исключительная важность включения в вычислительную систему на базе i486 очень быстрого дисковода. Считается, что для большинства применений использование жесткого диска емкостью менее 100 Мбайт практически сводит на нет преимущества процессора 80486 над кристаллом i386. Если же в вашем компьютере установлен 300-мегабайтный накопитель с контроллером SCSI или ESDI, можно быть уверенным, что i486 постарается проявить себя во всей красе: такие накопители характеризуются чрезвычайно малыми временами поиска и перехода с дорожки на дорожку, да и контроллер работает достаточно быстро, чтобы не испортить общую картину.

Стоит ли тратиться?

Итак, вы решились на покупку системы 486 и теперь раздумываете над такой мелочью, как тактовая частота. Читателям, наверное, известно, что на западном рынке можно увидеть и 25- и 33-мегагерцевые варианты, хотя первые пока еще встречаются куда чаще.

Позволю себе напомнить, что случилось в свое время с машинами на базе 80386. Буквально два года назад 16-мегагерцевые компьютеры являли собой арьергард в этом шустром семействе, но довольно скоро на этом незавидном месте оказались сначала 20-ти, а затем и 25-мегагерцевые аппараты. Покупая 486 машину с тактовой частотой 25 Мгц, вы рискуете через полгода оказаться в хвосте, а еще через полгода вам придется серьезно подумать над проблемой приобретения более быстрого компьютера. Я никому не советую на практике столкнуться с последствиями народной мудрости, которая, как известно, гласит: "скупой платит дважды", и рекомендую раскошелиться сейчас.

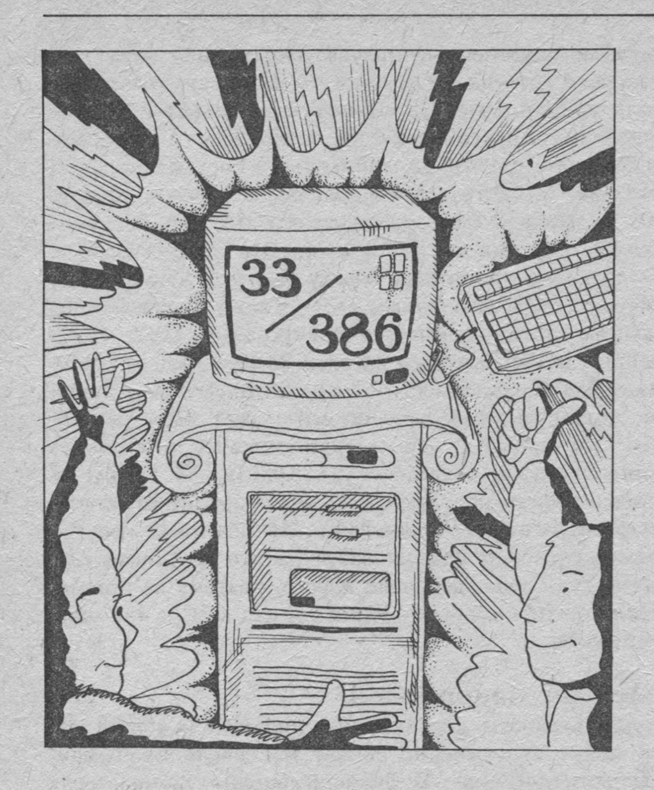
Решая вопрос, какой шинной архитектуре отдать предпочтение, следует иметь в виду, что будущее за двумя новыми стандартами — MCA и EISA. В отличие от 16-разрядных гнезд расширения, поддерживаемых старой шиной ISA, 32-разрядные гнезда стандарта EISA позволят грядущим периферийным устройствам заметно быстрее обмениваться данными с центральным процессором. Шина MCA вообще создавалась специально для работы в комплексе с i486 при скорости передачи информации 106 Мбайт в секунду. Но не стоит сбрасывать со счетов и тот факт, что старая 16-разрядная периферия значительно разнообразнее и дешевле новой, а тем более предназначенной для подключения к шине MCA. Так что окончательный выбор целиком за вами.

И все-таки, давайте попытаемся ответить на вопрос: а нужна ли лично вам система, построенная на базе i486?

В целом ряде приложений такая машина не будет иметь ощутимого преимущества над хорошо скомплектованным компьютером с процессором 80386. Если вы работаете в однопользовательском режиме и основными вашими пакетами являются текстовые процессоры, базы данных или электронные таблицы, покупка 386 машины сэкономит вам несколько тысяч долларов, а эффект от использования вычислительной техники будет аналогичным.

Компьютер на базе i486 незаменим при работе в такой многопользовательской операционной системе, как UNIX, особенно в тех случаях, когда программы написаны с учетом максимального использования преимуществ 32-разрядной системы команд. Вот тут вы немедленно почувствуете увеличение производительности раза в 3-4. Что касается издательских пакетов и САПР, серьезной графики и сложной статистики, то в этих сферах процессор 80486 просто недосягаем. Не следует забывать также такой важный слой, как локальные сети, где персональный компьютер на базе i486 нашел свое место в качестве мощного файл-сервера.

И.Липкин



Стоит ли вам покупать 33-мегагерцевый компьютер на базе i80386? Если вы серьезный пользователь, которому приходится работать с такими приложениями как САПР или настольное издательство, если вам необходим высокопроизводительный файл-сервер для локальной сети — время, которое позволит сэкономить 33-мегагерцевая система, полностью компенсирует материальные затраты.

PC 386/33
НА ЛЮБОЙ ВКУС

"В какое удивительное время мы живем!" — сказал персонаж одного из советских фильмов времен репрессанса. Я не берусь судить, что именно хотел выразить автор этой высокоидейной киноленты, вкладывая сии эпохальные слова в уста молодого строителя коммунизма, но именно эта фраза приходит мне на ум, когда я знакомлюсь с новейшей вычислительной техникой Запада.

Итак, в какое же удивительное время мы живем? За последние десять лет 16-разрядные микропроцессоры, которые поначалу использовались только в профессиональных вычислительных системах, "сползли" в бытовую электронику и того гляди вообще исчезнут из нашего поля зрения. Если так и дальше пойдет, Intel 80286, это чудо середины 80-х, через пару лет перестанут ставить даже в домашние компьютеры. Буквально каждые полгода ведущие фирмы пугают своих конкурентов то появлением сверхпроизводительного RISC-процессора с шиной данных аж в 128 разрядов, то супер-кристаллом, в который удалось втиснуть и центральный процессор и сопроцессор и кэш и еще Бог знает что. Если бы мне пару лет назад кто-нибудь сказал, что настольная "персоналка" может иметь производительность 17 миллионов команд в секунду — я бы отправил трепача проспаться. А сегодня я не удивлюсь сообщению о скорости, вчетверо превышающей эту цифру.

Но все равно — это пока еще экзотика (только надолго ли?). Более привычны нынче показатели поскромнее. Тем не менее, внимательно вчитайтесь в следующие слова: персональную вычислительную систему с производительностью 8-10 MIPS вполне реально приобрести за 4-5 тысяч долларов, причем это будет не "голый" системный блок, а нормально укомплектованная рабочая станция с супер-VGA монитором и жестким диском под 200 мегабайт. Звучит восхитительно, не правда ли?

Более подробно о PC-33

В то время как персональные компьютеры на базе процессора i80386 DX используются уже несколько лет, быстрые 33-мегагерцевые системы лишь недавно стали доступны пользователю со средними финансовыми возможностями. Наше желание приобрести лучшее из того, что позволяет бюджет, вполне естественно, но тут возникает вопрос "А нужно ли мне все это?".

На серьезную высокопроизводительную систему придется-таки раскошелиться. Посудите сами — средняя стоимость машины на основе микропроцессора 80386 SX составляет сегодня около 2 — 2,5 тысяч долларов. Цена же компьютерных "Феррари" и "Ломбарджини", включенных в наш обзор, варьируется от 3,5 до 15 тысяч долларов! Таким образом, повышение

производительности вам может обойтись дороже, чем сама система.

Но тут не надо упускать из виду, что существуют некоторые сферы применения компьютеров, где без дополнительных капиталовложений не обойтись. Наиболее ярким примером в этом отношении может служить использование машины в многозадачном режиме или в качестве файл-сервера сети с операционной системой OS/2 или UNIX. К тому же любое применение, связанное с графикой, только выиграет от перехода на тактовую частоту 33 МГц.

Необходимо особо упомянуть о том, что подобные системы отличаются от остальных не только своей высокой тактовой частотой и скоростью передачи данных по шине. Существует масса других факторов, которые также необходимо принимать во внимание. Все рассмотренные здесь компьютеры, как, впрочем, и большинство 33-мегагерцевых машин, имеющихся на рынке, для ускорения различных функций активно используют кэширование и возможность "затенения" ПЗУ BIOS и видео-ПЗУ в более быстрой оперативной памяти. Это позволяет максимально использовать преимущества быстрого центрального процессора.

Кэширование

Исторически, кэширование впервые стали применять для ускорения работы с магнитными дисками. Подобная модель кэш функционирует следующим образом: наиболее часто запрашиваемые с жесткого диска файлы запоминаются в какой-то одной определенной области ОЗУ. Получив команду чтения/записи с диска, процессор в поисках запрашиваемых данных сначала обращается к более быстрой оперативной памяти. Если нужная информация здесь имеется, дополнение, уничтожение или редактирование данных производится сразу, без обращения к самому диску. Если же в процессе работы программа управления кэш-памятью чаще обращается к диску, чем к ОЗУ, в область кэш автоматически переписывается та часть диска, к которой было больше обращений. Наименее используемая информация в этом случае переносится обратно на диск. Для того, чтобы при неожиданной неисправности системы как можно меньше информации было потеряно, через равные промежутки времени все содержание кэш-памяти переписывается на диск.

Еще один способ ускорения работы компьютера — это кэширование данных, содержащихся в различных ПЗУ системы (например, в ПЗУ BIOS или в видео-ПЗУ). Этот способ называется "затенением" (shadowing). Он существенно повышает производительность, так как время доступа в ОЗУ меньше, чем в ПЗУ. Наиболее эффективные методы затенения используют область оперативной памяти, расположенную в промежутке 640 Кбайт — 1 Мбайт. Таким образом удается эффективно использовать полный диапазон ОЗУ персонального компьютера.

Наконец, сам центральный процессор, несущийся галопом с тактовой частотой 33 МГц, работает гораздо быстрее самого "шустрого" динамического ОЗУ. Даже наиболее современные модули оперативной памяти со временем доступа 80 и даже 70 наносекунд заставляют процессор молотить вхолостую в ожидании окончания цикла запроса. Для того, чтобы свести время вынужденного простоя до минимума, в системе создается отдельная область сверхоперативной памяти, собранная на микросхемах статического ОЗУ со временем доступа, обычно, 20 наносекунд. Эта область используется в качестве буфера между быстрым центральным процессором и медленным ОЗУ. Такой процесс буферизации носит название кэширования ОЗУ (RAM caching) и используется во всех 33-мегагерцевых компьютерах. В зависимости от сферы применения машины, объем быстрой кэш-памяти может быть различным. Наименьший объем, применяемый в системах данного класса, составляет 32 Кбайта, 64 Кбайта — наиболее часто используемый объем, некоторые системы располагают 128 или даже 256 Кбайтами кэш-памяти. Однако, не следует забывать что 20-наносекундные модули стоят довольно дорого, поэтому не стремитесь истратить свои деньги на то, что вам не так уж и необходимо.

Быстрые накопители

Скорость работы дисковых накопителей и контроллеров также существенно влияет на общую производительность системы. В 33-мегагерцевых компьютерах наиболее часто применяются так называемые IDE-диски, являющиеся комбинацией накопителя и контроллера с интерфейсом SCSI или ESDI.

Сами накопители также претерпели некоторые конструктивные изменения. Мощные жесткие диски (80 Мбайт и более), оборудуются новым приводом магнитных головок, в основе которого лежит линейный двигатель. Он работает намного быстрее, чем привод, использующий обычный шаговый мотор.

Вообще, при проектировании машин на базе процессора i386 фирмами-изготовителями были использованы некоторые стандартные решения, среди которых, кроме выборочного затенения ПЗУ BIOS или видео-ПЗУ, можно назвать и двойную схему кэширования, и страничную организацию ОЗУ с чередованием и т.д.

В статье, которую, смею надеяться, вы прочитаете до конца, я попытался на нескольких примерах представить вам весь спектр компьютеров этого класса — от достаточно дорогих моделей до тех, что будут по карману и отечественному покупателю.

Итак, начнем с известных фирм, славящихся надежным и, что несколько хуже, дорогим оборудованием.

Фирма Advanced Logic Research
Модель ALR Flexcache 33/386DT

Flexcache 33/386DT — хорошо организованная вычислительная система, которая поддерживает 128 Кбайт 28-наносекундной кэш-памяти. Она может быть с успехом использована для настольного издательства или для автоматизации офиса (например, для ведения мощной базы данных). Эта первоклассная ма-

шина несколько дороговата по сравнению с другими системами, но ее отличает действительно отменное качество. При цене немногим более 7,3 тысяч долларов она является, пожалуй одной из лучших среди представленных на рынке 33-мегагерцевых систем.

К сожалению, цены на периферийное оборудование фирмы ALR "кусаются". Например, 33/386DT Model 10 — компьютер, идентичный рассмотренному в обзоре, но без жесткого диска — стоит на 2 тысячи долларов дешевле. Кстати, для ALR характерны довольно слабые возможности расширения внешней памяти: два гибких и один жесткий диск полной высоты могут занять все отведенное для этой цели пространство внутри системного блока.

Познакомимся поподробнее с одной из модификаций ALR Flexcache 33/386DT — системой FlexCache 33/386DT Model 120. Она выпускается с ОЗУ объемом 2 Мбайта, 1,2-Мбайтным пятидюймовым гибким диском и 120-Мбайтным жестким диском ESDI. Машина также оборудована последовательными и параллельным портами. Цена данного комплекта по каталогу — 6490 долларов. Если все это дополнить 14-дюймовым цветным монитором VGA стоимостью 499 долларов и соответствующим видеоадаптером, а это еще 329 долларов, то цена всей системы вырастет до 7318 долларов. Учтите, что установка 120-мегабайтного жесткого диска потребует 2000 долларов, а переход от 120 к 340 мегабайтам обойдется еще в 2000 долларов.

Существуют еще два варианта 33/386DT: Model 10 без жесткого диска (4490 долларов) и Model 340H (8490 долларов) с быстрым накопителем на жестком диске емкостью 340 Мбайт. Фирма ALR выпускает свои базовые конфигурации как в настольном исполнении типа low-profile, так и в виде стойки типа tower.

Все-таки основное, что определяет популярность 33/386DT — это качество. Распаковывая систему, вы, к примеру, с удивлением обнаружите небольшую легко снимающуюся пластиковую упаковку, закрывающую раму для установки дисководов. Это сделано не просто из эстетических соображений. Такое покрытие предохраняет зону накопителей от пыли, продлевая таким образом жизнь системе.

При всей внешней привлекательности, возможности расширения системы несколько ограничены. Как уже упоминалось, в ней предусмотрены только два места для 5,25-дюймовых флоппи-дисководов, и еще один отсек для устройства полной высоты.

Намного лучше обстоит дело с гнездами расширения. В компьютере их восемь, причем, одно 8-битное и семь 16-битных. Как правило, три 16-разрядных гнезда приходятся на видеоадаптер, плату последовательного/параллельного портов и контроллер дисков. При использовании оставшихся гнезд необходимо учитывать мощность источника питания — 200Вт.

Внутренний осмотр системы не преподнесет вам неприятных сюрпризов, несмотря на то, что часть материнской платы находится под блоком питания и направляющими отсека для накопителей. Это, очевидно, создаст некоторые неудобства при техническом обслуживании. Сама же материнская плата хорошо расположена, поверхность ее чистая, на ней нет лишних проводов и перемычек.

Быстрая кэш-память ALR 33/386DT размером 128 Кбайт существенно превышает привычный объем 32 или 64 Кбайта. Аналогично всем 33-мегагерцевым системам, которые будут рассмотрены ниже, компьютер располагает возможностью "затенения" ПЗУ BIOS. Тестирование машины показывает, что она может успешно справляться с приложениями, максимально насыщенными вычислениями. Если вы собираетесь эксплуатировать ее, в основном, как систему автоматизированного проектирования, я посоветовал бы вам нарастить ОЗУ и установить сопроцессор. При необходимости использования ALR 33/386DT в качестве файл-сервера, более удобной для вас будет Model 340H с ее 340-Мбайтным жестким диском.

Фирма American Mitac
Модель MPC4000G

Компьютер MPC4000G вызывает смешанные чувства. С одной стороны, он превосходно работает, имеет удобный корпус типа tower, в котором достаточно места для дополнительных накопителей и плат расширения. Руководство пользователя также представляется мне одним из лучших, написанных для подобных систем. С другой стороны, материнская плата очень напоминает полуфабрикат, кое как "доведенный" до нормального рабочего состояния: вся ее поверхность густо покрыта перемычками, а это, согласитесь, большой конструктивный недостаток.

Правда, представители American Mitac утверждают, что системы, которые поставляются на рынок сейчас, доработаны и все подобные "шероховатости" уже устранены. Кроме того, фирма готова заменить любую из старых материнских плат, и всегда придет на помощь пользователю, если у него возникли какие-либо проблемы.

Стандартная конфигурация компьютера Mitac включает в себя 4 Мбайта оперативной памяти и флоппи-дисковод высокой плотности (на выбор: 5,25-дюймовый или 3,5-дюймовый). Цена такой системы, включая MS-DOS и GW-BASIC — 4985 долларов. Дополнив комплект видеоадаптером VGA (199 долларов), очень красивым 14-дюймовым цветным монитором VGA, выпускаемым по заказу Mitac фирмой Tatung (485 долларов) и 104-мегабайтным жестким диском Conner IDE (725 долларов), вы получите работоспособную машину общей стоимостью 6394 доллара.

Поговорим о качестве и возможностях расширения компьютера. По многим показателям с этим у Mitac все в порядке. Корпус-стойка достаточно вместителен и, при желании, кроме, например, 3,5-дюймового флоппи-дисковода фирмы Panasonic и жесткого диска, каждый пользователь может установить в своей системе еще три накопителя со сменным носителем и

	Advanced Logic Research ALR Flexcfche 33/386DT	Advanced Logic Research ALR Microflex 3300	American Mitac MPC4000G	AST AST Premium 386/33	Argo Arche Arche Legacy 386-33	Computers Inc. Argo Tower 386/33
Цена системы ($)	7.318	14.006	6.394	10.245	12.285	3.642
Тип системного блока	настольный	башня	башня	настольный	настольный	башня
Размеры (см)	53.8x42.5 x15	57.5x18.8x46.3	21.3x47.5 x41.9	47.5x40 x16.3	52.5x40.6 x15.6	63.7x20x45
Объем ОЗУ (Мбайт)	2	4	4	2	8	4
Максимальный объем ОЗУ (Мбайт)	16	64	24	16	32	12
BIOS	Phoenix	Phoenix	Phoenix	AST	Arche	AMI
Тип и емкость жесткого диска (Мбайт)	ESDI 120	ESDI 642	ESDI 104	IDE 110	380 179	
Флоппи диск (Мбайт)	1.2	1.44	1.44	1.2 1.44	1.2 1.2 1.44	
Гнезда расширения	1—8 бит 7—16 бит	4—16 бит 3—32 бит 1—32 бит	8—16 бит	1—8 бит 2—16 бит 1—32 бит	6—16 бит 2—32 бит	2—8 бит 5—16 бит
Клавиатура (кол.клавиш)	101	101	101	101	101	101
Программное обеспечение, входящее в комплект	Спец. утилиты	DOS 4.01	MS-DOS 4.01 утилиты AST видео-драйверы	MS-DOS GW-BASIC	MS-DOS 3.3 GW-BASIC	DOS 4.1
Блок питания (Вт)	200	200	300	220	275	230
Сопроцессор	80387 Weitek	80387 Weitek	80387	80387 Weitek	80387 Weitek	80387

четыре винчестера половинной высоты. Мощность блока питания — 300 Вт — вполне достаточна для того, чтобы справиться с теми дополнениями, которые вы сочтете необходимыми.

Машина оборудована восемью 16-разрядными гнездами расширения. Два из них используются для видеоадаптера VGA и контроллера жесткого диска (контроллер гибкого диска находится на материнской плате). Крышка корпуса-стойки легко снимается, открывая легкий доступ ко всем гнездам. При таком просторе внутри системного блока машина врядли перегреется, даже будучи до отказа заполненной накопителями и периферийными платами.

О качестве монитора VGA, входящего в комплект системы, стоит сказать особо. Несмотря на то, что на нем стоит клеймо Mitac, этот мультисканирующий аппарат выпускается, как было уже отмечено, фирмой Tatung. Он показывает прекрасную разрешающую способность (800x600) и работает в режиме супер-VGA практически без искажений.

Компьютер MPC4000G сопровождается солидной документацией. В дополнение к общим справочникам по MS-DOS и GW-BASIC прилагается краткий справочник о дополнительных функциях DOS 4.01 и превосходное руководство пользователя. Этот том содержит очень хорошо выполненные иллюстрации, и изобилует инструкциями по усовершенствованию системы для тех пользователей, кто изъявит желание установить дополнительные накопители, периферийное оборудование или увеличить объем памяти. Фактически этот справочник мало чем отличается от тех, что входят в комплект поставки гораздо более дорогих компьютеров.

Существует, однако, два момента, на которые следует обратить внимание. Один из них не так важен — внутренняя деталь крепления корпуса пересекает материнскую плату и может затруднить ее замену в случае необходимости. Второе — я об этом уже говорил — это сама материнская плата, которая сплошь покрыта перемычками, многие из которых припаяны прямо к ножкам гнезд для микросхем. Перемычки на материнской плате, — явление не такое уж редкое, но

Automated Computer Tecnology Corp. **ACT386-33FC**	Brain Computer Corp. **The Brain 386-33**	CAF Technology Inc. **CAF Master 386C/33T**	Clone Computers **Clone 386-33C**	Compaq **Deskpro 386/33**	PC Brand **PC Brand 386/33**	Samsung **Samsung 386A3**
4.450	3.820	3.179	3.684	10.499	4.498	8.800
мини-башня	настольный	башня	мини-башня	настольный	настольный	настольный
39.5x16.9 x48.8	16.3x52.5 x42	61.3x18.8 x42.5	39.5x22x50	48x44.3x16.3	47.5x42.5 x15.6	53x41.8x16
8	4	4	4	2	1	4
8	16	8	8	16	16	16
AMI	AMI	AMI	AMI	Compaq	Phoenix	Samsung
IDE 143	IDE 160	ESDI 150	ESDI 160	84	80	130
1.2 1.44	1.2 1.44	1.2 1.44	1.2 1.44	1.2	1.2	1.2
1—8 бит 6—16 бит 1—32 бит	2—8 бит 6—16 бит	1—8 бит 5—16 бит 1—32 бит	1—8 бит 6—16 бит 1—32 бит	2—8 бит 6—16 бит	2—8 бит 6—16 бит	7—16 бит 1—32 бит
102	101	101	101	101	101	101
DOS 4.01	DOS 4.01	DOS 4.01 (3.3)	DOS 4.01 GW-BASIC утилиты	DOS 3.31	утилиты	-
200	200	200	200	-	145	375
80387 Weitek	80387 Weitek	80387	80387 Weitek	80387	80387 Weitek	80387

они обычно находятся на задней ее части и не припаяны непосредственно к ножкам. Принимая во внимание, что Mitac в настоящее время выпускает чистые платы, можно быть уверенным, что модель MPC4000G — стоящее приобретение с хорошими рабочими характеристиками.

Машина хорошо зарекомендовала себя в процессе тестирования. Обладая возможностями затенения ПЗУ BIOS и видео-ПЗУ, она могла бы использоваться как превосходный файл-сервер, или как мощный компьютер для настольной типографии.

Фирма Arche Legacy
Модель Arche Legacy 386-33

На первый взгляд, каталожная цена 12285 долларов должна отбить охоту знакомиться с системой Arche Legacy 386-33 у любого потенциального покупателя, находящегося в здравом уме и твердой памяти (кстати, эта цифра — далеко не предел для компьютеров данного класса). Однако, благодаря своей конфигурации типа "рабочая станция", что само по себе уже хорошо, и целому набору предлагаемых дополнений, машина отлично подходит для задач автоматизированного проектирования, сложнейших научных расчетов, для работы в качестве многопользовательской системы под управлением UNIX или мощного файл-сервера. Да, цена, конечно же, астрономическая, но не будем забывать, что компьютер DESKPRO 386/33 фирмы Compaq при той же самой конфигурации (за исключением того, что емкость жесткого диска там составляет 320 Мбайт, а не 380, как в Arche) стоит еще дороже. Кроме того, как показали результаты тестирования, модель фирмы Arche не то что не уступает, а даже превосходит остальные системы почти по всем показателям.

Правда, есть и другой вариант: "всего" за 7 тысяч долларов вы можете приобрести менее дорогую версию Legacy 386-33 без сопроцессора, с меньшим объемом оперативной памяти и жестким диском уменьшенной емкости. Но если бы мне предложили в подарок любую из описанных в обзоре систем, я, конечно же, выбрал бы Arche.

Давайте разберемся, из чего же складывается вся

мощь рассматриваемой нами рабочей станции. К базовой конфигурации (2 Мбайта ОЗУ и 5,25-дюймовый накопитель на гибком диске емкостью 1,2 Мбайта, цена 5606 долларов) добавляют 6 Мбайт 70-наносекундной оперативной памяти (1200 долларов), 33-мегагерцевый сопроцессор i80387 (600 долларов), 16-битный видеоадаптер VGA и цветной монитор VGA (450 и 640 долларов соответственно), контроллер жесткого диска ESDI (24 Мбайта/с) и жесткий диск емкостью 380 Мбайт (365 долларов и 3295 долларов соответственно), а также 3,5-дюймовый накопитель на гибком диске емкостью 1,44 Мбайт (129 долларов). Результирующая цена системы — 12285 долларов. Compaq DESKPRO 386/33, о котором речь впереди, имеет вполовину меньший объем ОЗУ, на четверть меньший жесткий диск и стоит на 2000 долларов дороже.

Как правило, компьютеры фирмы Legacy выпускаются в стандартном корпусе типа AT, тем не менее большинство моделей, включая 386-33, можно приобрести также еще в двух вариантах — в корпусе типа tower и в настольном исполнении типа low-profile. Но и стандартный AT-корпус представляется мне достаточно просторным для дальнейшего расширения системы. Здесь вполне хватает места для пяти накопителей половинной высоты, три из который могут иметь сменный носитель. В нашем случае в этом отсеке располагаются два флоппи-дисковода высокой плотности, диаметром 5,25 и 3,5 дюйма, и жесткий диск фирмы Maxstor полной высоты емкостью 380 Мбайт. При такой конфигурации остается свободным одно место, которое можно использовать для оптического накопителя, винчестера или стримера половинной высоты.

Внутренний осмотр системы показывает, что материнская плата удачно расположена, и на ней нет перемычек. Все остальные компоненты, входящие в состав компьютера, также достаточно высококачественные. Особенно приятно отметить, что в машине применены накопители на гибких дисках фирмы TEAC.

На материнской плате расположены восемь гнезд расширения: шесть 16-и и два 32-разрядных. Последние могут быть использованы для увеличения объема оперативной памяти до 16 Мбайт (что превышает максимальный для материнской платы размер ОЗУ вдвое), или для подключения 8-/16-битных контроллеров периферийных устройств. В нашей модели три из этих гнезд заняты адаптером VGA, платой последовательного/параллельного портов и платой комбинированного контроллера гибкого и жесткого дисков.

Как уже отмечалось, производительность машины выше всяческих похвал. Компьютер Legacy 386-33 превосходит подобную систему фирмы Compaq по результатам большинства тестов. Частично это можно объяснить наличием быстрой кэш-памяти, которая, в отличие от установленной на многих других машинах, имеет объем не 32 или 64 Кбайта, а целых 128 Кбайт.

Фирма AST
Модель AST Premium 386/33

В последнее время марка AST стала настолько широко известной, что это позволило компании здорово повысить цены на свои изделия. Решение, наверное, разумное, особенно если принять во внимание высокое качество продукции фирмы. Что же касается производительности, то в нашем конкретном случае ее трудно назвать выдающейся. Сугубо средние показатели при проведении дисковых тестов указывают на то, что Premium 386/33 — не самый лучший вариант для использования его в качестве файл-сервера. Однако общие результаты тестирования более чем удовлетворительны, в связи с чем система может быть рекомендована для создания на ее основе настольного издательства или рабочей станции САПР. К тому же при разработке машины была использована специальная архитектура, позволяющая быстро и сравнительно недорого переоснастить компьютер другим, более производительным, 25-мегагерцевым центральным процессором i486.

Существует несколько моделей AST Premium 386/33. Здесь мы рассмотрим Model 115V (цена 8495 долларов) с 5,25-дюймовым накопителем на гибком диске высокой плотности, ОЗУ объемом 2 Мбайта, видеоадаптером AST-VGA Plus и жестким диском IDE емкостью 110 Мбайт. Если система дополнена цветным монитором VGA, она стоит еще на 695 долларов дороже.

Высокое качество и интересные схемные решения фирмы AST производят очень хорошее впечатление. На поверхности материнской платы нет вызывающих раздражение дополнительных перемычек и проводов. Все компоненты системы подобраны со знанием дела: к примеру, жесткий диск емкостью 110 Мбайт поставляется фирмой Imprimis, флоппи-дисковод — компанией NEC, а блок питания мощностью 220 Вт изготовлен на заводе Astec.

Premium 386/33 обладает неплохими возможностями для расширения. В системном блоке можно разместить три 5,25-дюймовых накопителя половинной высоты со сменным носителем и два винчестера половинной высоты. В принципе, эти пять направляющих позволяют использовать дополнительные накопители любых размеров.

На материнской плате имеется достаточное количество гнезд для плат управления периферийными устройствами. Сюда входят: одно 8-битное, три 16-битных гнезда ISA и два 32-разрядных гнезда SmartSlot AST, которые могут быть использованы для расширения памяти или для подключения стандартных 8- или 16-разрядных периферийных плат. Дополнительный разъем SmartSlot предназначен для съемной платы процессора. На этой плате расположены центральный процессор и системное ОЗУ размером до 4 Мбайт. Такое конструктивное решение позволяет увеличить мощность системы, заменив имеющуюся плату на другую с процессором i486. Гнезда SmartSlots

AST обладают способностью распознавать тип установленной платы — периферийной 8- или 16-разрядной или 32-разрядной платы дополнительного ОЗУ. В последнем случае система подключает дополнительную память автоматически.

Документацию фирмы AST отличает высокое качество. В дополнение к общим справочникам по MS-DOS и GW-BASIC в комплект входит справочник по утилитам, в котором описаны все включенные в систему сервисные программы, а также хорошо иллюстрированный пользовательский справочник непосредственно по системе Premium 386/33.

Несмотря на то, что результаты тестирования Premium 386/33 вполне удовлетворительны, машине довольно далеко до самых быстрых из рассмотренных в нашем обзоре. Это несколько неожиданно, особенно если принять во внимание относительно малое время доступа к жесткому диску и возможность затенения ПЗУ.

Фирма Compaq Модель DESKPRO 386/33

Compaq DESKPRO 386/33 оказалась самой дорогостоящей из всех проанализированных машин. За те же 15 тысяч долларов можно приобрести две, а то и три системы, и при этом у вас еще останутся деньги на то, чтобы как следует отметить покупку в совместном советско-американском ресторане. Даже если отказаться от стримера и сопроцессора, компьютер все равно обойдется очень дорого.

Конечно, модель DESKPRO 386/33, как и все, что выпускает Compaq, отличается высочайшим качеством, но далеко не всегда оно окупается. DESKPRO 386/33 недосягаема по многим показателям, но по ряду параметров она не выдерживает конкуренции с машиной Arche Legacy 386/33, которая при аналогичной конфигурации стоит на несколько тысяч долларов дешевле.

Давайте посмотрим, откуда берется такая баснословная цена? Базовая система включает в себя: 2 Мбайта оперативной памяти, 5,25-дюймовый накопитель на гибком диске, жесткий диск емкостью 84 Мбайта и видеоадаптер VGA — все это уже стоит 10499 долларов. Флоппи-дисковод, диаметром 3,5 дюйма обойдется вам еще в 275 долларов. Добавьте сюда сопроцессор i80387 (1599 долларов), 2 Мбайта оперативной памяти (999 долларов), 60-Мбайтный накопитель на магнитной ленте (799 долларов), 14-дюймовый цветной монитор VGA (699 долларов) и DOS 3.31 (120 долларов). Итого 14990 долларов. Вот это да!

О том, что качество изделий фирмы Compaq всегда превосходно, известно даже нашему неискушенному пользователю. Для того, чтобы еще раз убедиться в этом, достаточно осмотреть системный блок внутри. Некоторые компоненты, например, блок питания, специально выпускается по заказу этой фирмы. К тому же многое Compaq выпускает и самостоятельно. К подобным "родным" устройствам относятся материнские платы и даже ПЗУ BIOS. Фактически, только плата видеоадаптера и жесткий диск пришли в систему DESKPRO 386/33 со стороны.

Системная плата компьютера имеет 8 гнезд расширения: два 8-разрядных и шесть 16-разрядных. В нашем случае свободными оказываются только четыре. Остальные заняты платой дополнительной памяти, контроллером гибкого диска, адаптером VGA и платой управления стримером. Несмотря на то, что предельную мощность блока питания фирма, как правило, не обозначает, репутация Compaq позволяет смело предполагать, что расширение системы не вызовет его перегрузки.

В дополнение к трем 5,25-дюймовым накопителям половинной высоты со сменным носителем, которые могут быть установлены со стороны передней панели системного блока, имеется еще два места для установки 3,5-дюймовых накопителей вблизи задней стенки корпуса.

Интересно, что 5,25-дюймовые флоппи-дисководы, выпускаемые фирмой, снабжены кнопкой защиты записи. С подобной особенностью мне пришлось встретиться впервые.

Документация, сопровождающая систему, просто превосходна. Текст хорошо составлен и разумно расположен, содержит множество графиков и иллюстраций. Помимо руководства пользователя, в документацию входят справочники по MS-DOS, GW-BASIC и утилитам Compaq.

По многим параметрам система показывает отличные результаты, но чаще всего она оказывается на втором месте после Arche. В общем, как и следует ожидать, Compaq работает хорошо, но не намного лучше, чем большинство других более скромных систем.

Фирма PC-BRAND Модель PC-BRAND 386/33

PC-BRAND 386/33 — система, которая характеризуется высоким качеством и вполне умеренной ценой. Если учесть, что в нее входят два накопителя на гибком диске, цветной монитор, 4 Мбайта оперативной памяти и жесткий диск емкостью 200 Мбайт, остается только удивляться, почему она стоит всего 4498 долларов. Конечно, вы можете приобрести компьютер, который имеет несколько лучшую документацию и работает чуть побыстрее, но вам придется заплатить за это куда больше (см. выше).

Стандартный комплект PC BRAND 386/33 стоит 2299 долларов и включает в себя 1 Мбайт оперативной памяти, 16-разрядную плату VGA, а также накопитель на гибком диске высокой плотности (5,25- или 3,5-дюймовый). При наличии жесткого диска емкостью 200 Мбайт и цветного монитора VGA цена поднимается до 4089. Увеличение объема оперативной памяти на 3 Мбайта обойдется еще в 304 доллара, а второй накопитель на гибком диске — в 105 долла-

ров. Вот мы и получили итоговую сумму — 4498 долларов.

Поверхность материнской платы компьютера лишена перемычек и дополнительных проводов, а компоненты, входящие в его состав, например, жесткий диск емкостью 200 Мбайт, удовлетворят своим качеством любого покупателя.

Система размещается в стандартном корпусе типа AT и располагает вполне приличными возможностями для расширения. Помимо трех накопителей половинной высоты диаметром 5,25 дюйма и одного 3,5-дюймового, к которым возможен доступ извне, имеется еще место для установки двух дополнительных винчестеров. В нашем случае система оборудована 5,25- и 3,5-дюймовыми флоппи-дисководами, а также 3,5-дюймовым жестким диском. При такой конфигурации остаются свободными два места для монтажа 5,25-дюймовых накопителей со сменным носителем и еще два места для жестких дисков, а это неплохие условия для дальнейшего расширения системы.

На основной плате находятся восемь гнезд расширения — два 8-битных и шесть 16-битных. В нашем случае два 16-битных гнезда заняты видеоадаптером и комбинированной платой ввода/вывода и контроллера жесткого/гибкого диска. При использовании достаточно емких модулей оперативной памяти, на материнской плате можно уместить до 16 Мбайт ОЗУ.

Единственное, что заслуживает критики, это блок питания мощностью 145 Вт. Этого конечно же совершенно недостаточно для того, чтобы использовать все возможности расширения, заложенные в системе.

Что касается быстродействия, то здесь потенциального покупателя не ожидает никаких неприятных сюрпризов. Будучи не самой быстрой, PC BRAND, тем не менее, при тестировании показывает превосходные результаты. Использование возможностей кэширования и затенения существенно повышает общую производительность системы.

Отличное быстродействие и умеренная цена делают PC BRAND великолепной настольной машиной, с помощью которой вполне можно решать любые задачи. При той конфигурации, которую мы здесь рассмотрели (то есть с жестким диском емкостью 200 Мбайт и четырьмя мегабайтами оперативной памяти) она может быть использована и как превосходный сетевой файл-сервер.

Фирма Samsung
Модель Samsung 386/A3

Фирма Samsung уже давно выпускает компьютеры, хотя по своей популярности она и уступает другим маркам. Кстати, компания Novell продавала свои сети вместе с программным обеспечением NetWare, а файл-серверы для этих сетей были выпущены фирмой Samsung. Сервер A3 — это 33-мегагерцевый компьютер на базе i80386, специально созданный для использования в сетях Novell. Но согласитесь, когда в системе установлен быстрый жесткий диск достаточно большой емкости, как, например, 134-мегабайтный накопитель ESDI в анализируемой модели, ее вполне можно использовать как для сетевых целей, так и любых других приложений, требующих обработки больших массивов данных.

Система обладает великолепными возможностями расширения и показывает очень высокие результаты во время тестирования. К сожалению, она не относится к числу недорогих, так как даже без монитора и жесткого диска стоит уже более 6000 долларов. Базовый комплект 386/A3 стоимостью 6495 долларов содержит 4 Мбайта оперативной памяти и 5,25-дюймовый флоппи-дисковод емкостью 1,2 Мбайт. Цена видеоадаптера супер-VGA составляет 299 долларов, монитора — 699 долларов. Если к этому прибавить отформатированный жесткий диск емкостью 130 Мбайт — еще 1295 долларов, — то итоговая цена составит 8800 долларов.

Samsung A3 размещается в корпусе типа AT несколько увеличенных размеров. Объяснение тут очень простое: фирма с самого начала предусматривала возможности для расширения, задумав использовать систему в качестве файл-сервера. Внутри корпуса можно разместить три 5,25-дюймовых накопителя половинной высоты со сменным носителем и еще два устройства половинной высоты типа винчестер.

На материнской плате расположены одно 32-разрядное и семь 16-разрядных гнезд расширения. Как правило, два 16-битных гнезда заняты видеоадаптером VGA и комбинированной платой Western Digital WD1007A-WA2, содержащей контроллеры жесткого/гибкого дисков и электронику портов ввода/вывода. Единственное 32-разрядное гнездо предназначено для расширения ОЗУ свыше 8 Мбайт.

Разработчики A3 позаботились о том, чтобы у пользователей не возникало никаких проблем с подключением дополнительного оборудования: блок питания обладает исключительной для машин подобного класса мощностью — 375 Вт.

По результатам тестирования Samsung 386/A3 занимает почетное место среди самых быстродействующих моделей данного обзора.

Фирма Argo Computers
Модель Argo Tower 386/33

Вот тут-то и начинается самое интересное — вычислительные системы дешевле пяти и даже четырех тысяч долларов.

Компьютер представляет собой вертикальную стойку с размерами 63,8x20x45 см. Давайте посмотрим, достаточно ли плотно она забита аппаратурой. Базовая конфигурация стоимостью 3534 доллара включает в себя ОЗУ объемом 4 Мбайта, 32-Кбайтную кэш-память, жесткий диск фирмы Connor, который может быть сформатирован как на 179, так и на 212 Мбайт, флоппи-дисковод фирмы TEAC емкостью 1,2 Мбайта и видеоадаптер, поддерживающий разрешение 1024x768 точек (в видеоплате использован набор мик-

росхем Tseng ET4000). В комплект также входит 14-дюймовый цветной супер-VGA монитор Sony MultiScan, обеспечивающий максимальное разрешение 1024x768 при чересстрочной развертке и, конечно же, полноразмерная клавиатура. Компьютер оборудован одним параллельным и двумя последовательными портами.

Теперь обратим свое внимание на системную плату. Помимо корпусов процессора и BIOS вы обнаружите здесь универсальное гнездо для подключения сопроцессора: оно подходит не только для микросхемы i387, но и для Weitek 3167. Расположенная тут же кэш-память (объем которой, к сожалению, не может быть увеличен) реализована на 20-наносекундных кристаллах и управляется контроллером i82385. Максимальный объем ОЗУ — 8 Мбайт — на материнской плате может быть набран из 80-наносекундных модулей SIMM по одному мегабайту каждый или из 256-Кбайтных микросхем. При использовании платы расширения памяти результирующий размер ОЗУ составляет 12 Мбайт.

Очень удобно то, что органы управления компьютером расположены в верхней части передней панели стойки. Здесь можно увидеть тумблер включения компьютера, кнопки турбо-режима и сброса, а также комбинированный замок, запирающий клавиатуру или всю систему.

Отсек накопителей позволяет разместить от четырех устройств полной высоты до восьми устройств половинной высоты в любой комбинации, причем 4 верхних могут быть накопителями со сменным носителем.

Не забыть бы о гнездах расширения. Всего их восемь: два 8-разрядных, пять 16-разрядных и одно 32-разрядное для подключения дополнительной платы ОЗУ (его можно также использовать и как стандартное 8-разрядное гнездо).

Не правда ли, все, о чем было сказано выше, выглядит слишком солидно для системы, которая стоит чуть больше 3,5 тысяч долларов. Возникает резонный вопрос: а нет ли тут какого подвоха? Отвечаю — есть, причем несколько, но даже все они в сумме не определяют, на мой взгляд, столь низкую стоимость компьютера. Первый и совершенно непонятный сюрприз ожидает пользователя, желающего увеличить возможности своей машины: 230-ваттный блок питания оснащен всего четырьмя кабелями для подключения накопителей. К тому же довольно высокая стойка оборудована всего одним вентиллятором, которого едва хватает для охлаждения базовой конфигурации. Два эти момента следует иметь в виду, если вы захотите использовать Argo Tower 386/33, например, в качестве файл-сервера — дооснащение повлечет за собой проблемы, связанные с подводом энергии и отводом тепла.

И, наконец, несколько слов о производительности. Тестирование компьютера показывает не самые высокие, но и далеко не самые низкие результаты. Практически он занимает место как раз посередине диапазона в своем классе аппаратуры с точки зрения работы процессора, ОЗУ и системного ввода/вывода. Что же касается видеоподсистемы, то здесь Argo Tower 386/33 явно находится в числе лидеров.

Фирма Automated Computer Technology Модель ACT386-33FC

Стойка типа mini-tower (мини-башня), в отличие от стоек типа tower, как правило, предназначается для установки на столе, и если в случае полноразмерной стойки флоппи-дисководы расположены в верхней части корпуса, то в мини-башне все накопители сгруппированы внизу, таким образом, чтобы щель самого верхнего из них приходилась как раз посередине передней панели корпуса. Что же мы увидим внутри аккуратной мини-башни ACT, кроме 1,2-Мбайтного дисковода фирмы Chinon? Если перед нами базовая конфигурация стоимостью 4450 долларов, то, вскрыв корпус, мы обнаружим ОЗУ объемом 4 Мбайта, 64-Мбайтную кэш-память и жесткий диск Seagate/CDC/Imprimis емкостью 167 Мбайт. Видеоподсистема компьютера состоит из супер-VGA адаптера PowerGraph, поддерживающего разрешение 1024x768 точек при чересстрочной развертке и цветного супер-VGA монитора CTX CSV-3450. В видеоадаптере использован тот же набор микросхем Tseng ET4000, что и в компьютере фирмы Argo Computers. Картину довершают два последовательных и один параллельный порт, а также клавиатура, содержащая 102 клавиши.

На материнской плате машины, кроме процессора, гнезда для i387 или Weitek 3167 и ПЗУ системного ввода/вывода расположены корпуса кэш-памяти процессора с контроллером i82385 и модули ОЗУ. Кэш набирается из 20-наносекундных кристаллов и может достигать максимум 256 Кбайт. Что же касается ОЗУ, то тут все сделано в принципе так же как и в компьютере Argo Tower 386/33, за исключением одной "мелочи": при помощи платы расширения памяти ее объем можно увеличить до 16 Мбайт.

Стойка компьютера (39,5x17x48,8 см) закрыта спереди дверцей, за которой находятся все органы управления и приемные щели флоппи-дисководов. Снаружи остались только тумблер включения питания и индикатор тактовой частоты. Конструкция предусматривает размещение внутри корпуса пяти накопителей половинной высоты, причем все они могут иметь сменный носитель. Блок питания мощностью 200 Вт сможет обеспечить энергией все внутренние дополнительные устройства, так как силовых кабелей как раз тоже пять.

Из восьми гнезд расширения одно — 8-разрядное, шесть — 16-разрядных и одно — 32-разрядное для подключения платы расширения памяти.

По результатам тестов ACT386-33FC занимает позиции явно выше среднего уровня. Исключение, пожалуй, составляет ОЗУ, имеющее средние показатели. Интересно, что проверка функционирования видео-BIOS ставит компьютер на одно из последних мест в списке аппаратуры данного класса, а сам адаптер ра-

ботает настолько быстро, что "вытягивает" всю видеоподсистему по суммарной производительности в первые ряды.

Фирма Brain Computer
Модель Brain 386-33

Эта машина обращает на себя внимание прежде всего солидными результатами тестирования и высоким качеством своих компонентов.

В отличие от двух предыдущих компьютеров, Brain 386-33 — это более привычный для советского пользователя традиционный настольный аппарат. Давайте поближе познакомимся с его начинкой. За 3820 долларов фирма оснащает свой компьютер 4-Мбайтным ОЗУ, 64-Кбайтной кэш-памятью, жестким диском Seagate/CDC/Imprimis емкостью 160 Мбайт со временем доступа 16 мс и парой флоппи-дисководов высокой плотности (5,25 и 3,5 дюйма). Видеосистема включает в себя адаптер Tseng и цветной супер-VGA монитор NEC 2A с разрешением 800x600 точек. В комплект также входит полноразмерная (101 клавиша) клавиатура, один параллельный, один игровой и два последовательных порта.

На системной плате расположен набор микросхем OPTI 386, в состав которого, кроме процессора, входит также и контроллер кэш-памяти. Сама кэш, состоящая из 25-наносекундных кристаллов, может быть расширена максимально до 128 Кбайт. Здесь же, прямо на материнской плате, может быть размещено до 16 Мбайт ОЗУ, причем общая производительность системы почти не зависит от того, какие стоят кристаллы — 60- или 80-наносекундные: использование кэш-памяти и страничной организации ОЗУ с чередованием сводит на нет разницу во времени доступа. Организация BIOS принципиально ничем не отличается от описанных выше.

Внутри системного блока (16,25x52,5x42 см) достаточно места для размещения пяти накопителей половинной высоты, три из которых могут иметь сменный носитель, правда кабелей питания всего четыре, так что выпутываться из этой хитрой арифметики придется в конце концов покупателю. Блок питания компьютера рассчитан на выходную мощность 200 Вт.

Среди гнезд расширения мы не найдем 32-разрядного, потому что машине не требуется дополнительная плата памяти, а остальных восьми (двух — 8-разрядных и шести — 16-разрядных) вполне достаточно для удовлетворения потребностей самого привередливого пользователя.

Фирма CAF Technology
Модель CAF Master 386C/33T

Эта машина вызовет повышенный интерес у советского покупателя, для которого цена, да еще в валюте, играет, как мне кажется, далеко не последнюю роль.

Представьте себе, что солидный аппарат типа tower с супер-VGA монитором и 150-Мбайтным жестким диском стоит "всего" 3079 долларов. Увидев такую цифру многие покупатели просто не обратят внимания на довольно посредственные результаты тестов.

Итак, о комплектации. Базовая конфигурация содержит, кроме жесткого диска и монитора фирмы ADI с разрешением 1024x768, 70-наносекундное ОЗУ объемом 4 Мбайта, 64-Кбайтную кэш со временем доступа 25 наносекунд, пятидюймовый флоппи-дисковод высокой плотности и видеоадаптер Hisland. С внешним миром компьютер может общаться посредством параллельного, игрового и двух последовательных портов. На системной плате, кроме обычного набора корпусов центрального процессора, ПЗУ BIOS, кэш-памяти и гнезда для сопроцессора, можно обнаружить 8 гнезд для подключения мегабайтных модулей SIMM.

Стойка с линейными размерами 61,3x18,8x42,5 см сконструирована таким образом, что все шесть возможных внутренних накопителей половинной высоты могут использовать сменный носитель. Это очень разумно, особенно если учесть, что рынок с каждым годом все больше заполняется оптическими запоминающими устройствами большой емкости.

Гнезд расширения на системной плате всего семь, причем одно из них — 8-разрядное, одно можно использовать и как 16- и как 32-разрядное для расширения ОЗУ, все остальные — 16-разрядные. Заполнив их видеоадаптером, контроллером дисков, платой параллельного/последовательных портов, дополнительной платой памяти и платой мыши, мы обнаружим, что осталось всего два пустых гнезда. Но повторяю, все это мелочи по сравнению с впечатляюще низкой ценой компьютера CAF Master.

Фирма Clone Computers
Модель Clone 386-33C

Здесь мы еще раз встречаемся с настольным вариантом вертикальной стойки. Mini-tower фирмы Clone с линейными размерами 39,5x22x50 см оказывается достаточно вместительным для шести устройств половинной высоты. Конструкторы ухитрились в дополнение к традиционным направляющим, предназначенным для монтажа "стопы" из пяти накопителей, втиснуть внутрь корпуса еще одну кассету для крепления дисковода: она располагается параллельно материнской плате, но у противоположной стенки корпуса. Таким образом, платы расширения системы находятся как раз между материнской платой и этой кассетой. Остановимся попутно на гнездах расширения. Их всего восемь: одно 8-разрядное, шесть 16-разрядных и еще одно, которое можно использовать и для 32-разрядной платы расширения ОЗУ и в качестве 8-разрядного гнезда.

Но я несколько забежал вперед, вернемся к стандартной конфигурации. Система стоимостью 3763 доллара содержит 4 Мбайта ОЗУ, набранного из 70-наносекундных модулей, 64 Кбайта 25-наносекундной кэш-памяти, жесткий диск фирмы Imprimus емкостью 160 Мбайт, два флоппи-дисковода высокой плотности

(3,5 и 5,25 дюйма), параллельный, игровой и два последовательных порта. Видеоподсистема компьютера с разрешающей способностью 1024x768 точек включает в себя супер-VGA адаптер фирмы STB Systems и цветной монитор Liton.

На материнской плате нас не ждет ничего экзотического: традиционная комбинация набора микросхем и гнезд для восьми мегабайтных модулей SIMM.

Вообще, система оставляет очень приятное впечатление. Не говоря о том, что дизайнеры постарались придать самой стойке нестандартную стильную форму, по результатам тестирования Clone 386-33C при своей более чем скромной цене почти не знает себе равных.

До сих пор мы рассматривали вычислительные системы, основанные на использовании привычной шинной архитектуры EISA. Но "под занавес" мне хотелось бы познакомить вас с альтернативным направлением развития компьютерной индустрии.

Итак, внимание, перед вами шедевр микроканальной архитектуры —

ALR MicroFlex 3300, созданный специалистами уже знакомой нам фирмы **Advanced Logic Research.**

Стандартная конфигурация машины стоимостью 7909 долларов выпускается в виде настольного системного блока, в состав которого входит ОЗУ размером 4 Мбайта, кэш-память 128 Кбайт, жесткий диск емкостью 120 Мбайт с ESDI-контроллером и 3,5-дюймовый флоппи-дисковод. Видеосистема состоит из 16-разрядного видеоадаптера и цветного монитора VGA. Естественно, имеются последовательный и параллельный порты, а также порт для мыши.

Но это — минимум. Заплатив приблизительно в два раза больше, вы можете стать обладателем этакого праздничного комплекта в виде полноразмерной башни, внутри которой вместо хилого 120-Мбайтного накопителя будет стоять монстр емкостью аж в 642 Мбайта! Скорость у этого гиганта фирмы Maxtor тоже гигантская — время доступа равно 16 Мс. Для управления таким бронтозавром в компьютере использован ESDI-контроллер, работающий с тактовой частотой 15 МГц и оборудованный своей кэш-памятью в 32 Кбайта. Ну и для полного букета сюда еще добавили 33-мегагерцевый сопроцессор i387.

В машине MicroFlex 3300 фирма ALR использовала материнскую плату собственной конструкции с ПЗУ BIOS фирмы Phoenix и контроллером кэш-памяти i82385. Для ускорения работы оперативной памяти для системы создана уникальная шина данных удвоенной ширины (64 разряда), соединяющая основное ОЗУ с 64-Кбайтной кэш-памятью. Единственная плата памяти, на которой устанавливаются 4-Мбитные модули, может содержать максимально до 64 Мбайт ОЗУ.

Заглянем внутрь корпуса-башни. Здесь предусмотрены направляющие для размещения четырех накопителей половинной высоты, три из которых могут иметь сменный носитель, и еще одно место для винчестера полной высоты. Двухсотваттный блок питания оборудован достаточным количеством кабелей для подключения максимального числа устройств.

Теперь о гнездах расширения. Когда в системе установлены плата памяти, контроллер диска и видеоадаптер, незанятыми остаются еще четыре гнезда: два 32-разрядных и два 16-разрядных, причем два последних могут быть использованы для подключения дополнительных видеоадаптеров.

Несмотря на то, что MicroFlex 3300 — машина очень быстрая и перспективная, сравнительный анализ показывает ее относительно низкую конкурентоспособность на отечественном рынке. И дело тут не в качестве, к которому не придерешься, а в цене. Ну куда ей тягаться с Clone 386-33C или CAF Master 386C/33T! Так что с приобретением дорогих компьютеров нам скорее всего придется пока подождать. А когда появятся деньги, я думаю, придет время других машин, построенных на базе 50- или 65-мегагерцевых i80586 или RISC-процессоров.

И.Липкин

По материалам:

W.Rosch "Cashing In on the Micro Channel", PC Magazine, October, 1990.
A.Poor "33-MHz 386s: Mainstream Muscle", PC Magazine, December, 1990.

Фирма Spark International Inc. выпустила беспроводную мышь и Trackball. Оба устройства передают информацию о перемещении в инфракрасном диапазоне. Приемник излучения устанавливается как на клавиатуре Macintosh, так и подключается к его порту. В режиме мощного сигнала устройство сможет работать на удалении до 5 метров от компьютера. Оба устройства совместимы с Macintosh, PC, Amiga и Atari ST. Они поставляются со стандартным интерфейсом RS-232 для PC и ADB для Mac'a.

Spark Cordless Trackball будет продаваться за 185 долларов. В комплект входят собственно устройство, инфракрасный приемник, четыре батарейки и руководство пользователю. Мышь стоит на 10 долларов меньше.

Newsbytes News Network, December 20, 1990

Фирма Apple пытается поддержать конкурентоспособность своего портативного компьютера за счет уменьшения его размеров и цены. Как сообщает газета San Francisco Chronicle, на открывающейся 10 января выставке MacWorld Expo может быть продемонстрирован новый портативный компьютер фирмы. Машина будет весить 10 фунтов (4.5 кг) и стоить 4000 долларов — на 4 фунта (1.8 кг) легче и на 1500 долларов дешевле, чем нынешняя модель. Фирма также разрабатывает компьютер-записную книжку, который должен быть готов к августу, и карманную модель Macintosh — ее поступление в продажу планируется в следующем году.

Newsbytes News Network, December 20, 1990



В этом номере "КомпьютерПресс" мы начинаем цикл статей об архитектуре микропроцессоров семейства 80x86.

АРХИТЕКТУРА ПРОЦЕССОРОВ 80x86

Немного истории

Появление первых микропроцессоров можно смело назвать эпохальным событием второй половины XX века, которым мы обязаны фирме Intel Corporation of Santa Clara, что в знаменитой Кремниевой долине в Калифорнии.

Все началось в 1971 году, когда фирма создала микросхему 4004, сумев разместить в одном кристалле бóльшую часть компонентов процессора. Раньше подобное устройство представляло собой весьма объемистый блок. На базе 4004 был сделан первый микрокалькулятор, по тем временам казавшийся чудом техники. Почти в то же время появился специализированный микропроцессор 8008, который предназначался для использования в терминале вычислительной машины. Несмотря на то, что эти микросхемы были встречены без всеобщего восторга, видимо из-за их высокой стоимости и традиционной инертности мышления производителей электронной аппаратуры, фирма Intel продолжила свои работы в этой области, и после объявления в 1974 году о выпуске микропроцессора второго поколения 8080, компьютерная промышленность наконец зашевелилась.

Intel 8080 был первым универсальным устройством, предназначенным не только для замены жесткой логики, но и для вычислений. Он имел довольно высокую производительность, позволял адресовать значительный объем памяти, да и спектр его возможностей был существенно шире, чем у предшественников. Все это, очевидно, послужило причиной того, что микропроцессор быстро стал общепризнанным стандартом, и многие фирмы начали выпускать его по лицензии. Стали появляться улучшенные версии, например, Z80 фирмы Zilog или V10 фирмы NEC, но структура оставалась прежней. Кстати, и сама фирма Intel в 1976 году тоже выпустила модернизированный вариант кристалла 8080,— микропроцессор 8085 — заметно улучшенный аппаратно и, кроме того, дополненный несколькими командами. В него был встроен тактовый генератор и контроллер шины, добавлен простой последовательный порт и увеличена тактовая частота.

В 1978 году появилось третье поколение микропроцессоров, и опять фирма Intel оказалась лидером — кристалл 8086 стал первым микропроцессором, оперирующим 16-разрядными словами данных. Он обладал весьма высокой по тем временам производительностью и серьезными возможностями, в том числе полной десятичной арифметикой, что позволяло применять его в самых различных областях.

Здесь хотелось бы упомянуть, что фирма Intel неоднократно делала, как сказали бы военные, "маневры тактического отступления", приводившие ее к очеред-

В зависимости от набора функциональных узлов и внутренней организации микропроцессора выделяют три основных варианта: однокристальные микропроцессоры, многокристальные секционные микропроцессоры и однокристальные микро-ЭВМ.

Для однокристальных микропроцессоров характерны фиксированная разрядность и фиксированный набор команд. Кристалл содержит, как правило, арифметико-логическое устройство, блок дешифрации команд, устройство управления, устройство управления внутренними узлами и обменом информацией с внешними устройствами, а также буфера шин, согласующие внешние сигналы с внутренними.

Зачастую, в однокристальных микропроцессорах используется совмещенная шина данных и адреса, что объясняется небольшим количеством выводов корпуса микросхемы (обычно 40-48). Устройство управления реализует определенный набор команд, выдавая соответствующие управляющие сигналы различным узлам и внешним устройствам. Набор команд (фактически – логика работы устройства управления) реализован аппаратно и не может быть изменен разработчиком вычислительного устройства.

В многокристальных микропроцессорах используется несколько микросхем, каждая из которых реализует отдельный узел процессора. Как правило такими узлами являются секции АЛУ, блок микропрограммного управления, специальное постоянное запоминающее устройство, содержащее команды, представляющие собой последовательности микроинструкций процессора, а также узлы, обеспечивающие работу микропроцессора. Обычно секции АЛУ выпускаются с определенной разрядностью, но возможно параллельное соединение секций, что дает возможность получения вычислительных систем практически любой разрядности. Такие микропроцессоры применяются в мощных и быстродействующих устройствах, в частности, предназначенных для управления сложными объектами, производствами и т.д.

Однокристальные микро-ЭВМ отличаются тем, что в одной БИС реализован не только микропроцессор, но и другие узлы микроЭВМ, обычно ОЗУ, память программ, тактовый генератор, несложный контроллер прерываний, последовательные и/или параллельные порты, таймеры и т.д. Существуют специализированные однокристальные микроЭВМ для обработки аналоговых сигналов, содержащие еще и многоканальные аналогово-цифровые и цифроаналоговые преобразователи. Однокристальные микроЭВМ – наиболее универсальные устройства, однако обычно их производительность невелика, поэтому они используются в несложных управляющих системах, в измерительных приборах, в бытовой технике, в электронных игрушках и других подобных вещах.

И.Вязаничев

ному успеху. Например, в 1979 году она упростила процессор 8086, создав микросхему 8088 с 8-разрядной внешней шиной данных. Двумя годами позже фирмой IBM на основе этого кристалла был создан один из первых "настоящих" персональных компьютеров, и всемирное триумфальное шествие микропроцессоров фирмы Intel началось.

Другим "отступлением" явилось создание однокристальной микро-ЭВМ 8748, объединившей процессор, тактовый генератор, контроллер шины, постоянное запоминающее устройство для хранения программ, маленькое ОЗУ и два дополнительных параллельных порта — и все это в единственной микросхеме. Возможности этого устройства были сильно ограничены — объем оперативной памяти составлял мизерную величину в 64 байта, да и адресуемое пространство программ и количество внешних устройств также были небольшими. Правда, несмотря ни на что, 8748 вместе с еще более простым вариантом — 8035 — быстро завоевали рынок несложных устройств управления.

Но вернемся к семейству 8086. В 1983 году Intel разработала еще два микропроцессора, представлявших собой усовершенствованные варианты 8086 и 8088 — 80186 и 80188, однако получить широкого распространения они не успели, так как в том же году появился процессор 80286, ставший серьезным шагом вперед. Всего через год на его базе был создан персональный компьютер IBM PC/AT, предоставивший в распоряжение пользователя вычислительные мощности средней ЭВМ. С появлением виртуального режима стало возможным создавать на базе 80286 системы с разделением ресурсов, что раньше было прерогативой больших машин. В микропроцессоре было также реализовано управление памятью.

О более поздних моделях процессоров фирмы Intel мы попытаемся рассказать в следующих статьях, посвященных их архитектуре.

Архитектура микропроцессоров

Под этим термином понимают совокупность и способ объединения узлов микропроцессора, а также его на-

бор команд. Знание этих двух моментов дает возможность грамотно организовать интерфейс аппаратных и программных средств вычислительной системы. Считается, что минимальная архитектура микропроцессора требует наличия арифметико-логического устройства, выполняющего все операции преобразования поступающих данных, и устройства управления, обеспечивающего выполнение команд процессора и работу с внешними устройствами.

Кроме того, в микропроцессоре обязательно используются шины. Шина — это совокупность линий, по которым передаются цифровые сигналы, необходимые для обмена информацией между устройствами. В микропроцессорах фирмы Intel выделяют три шины: шину данных, шину адреса и шину управления. Кроме того, под шиной может подразумеваться стандартный набор линий, объединяющий в себе все эти три группы.

АРХИТЕКТУРА МИКРОПРОЦЕССОРА i8080

8080 является однокристальным микропроцессором, работающим с 8-разрядной шиной данных и 16-разрядной шиной адреса. Управляющие сигналы передаются по шине управления. Шины отделены друг от друга. Структура Intel 8080 приведена на рис. 1.

Микропроцессор содержит внутреннюю шину данных, посредством которой происходит обмен информацией между внутренними регистрами, арифметико-логическим устройством, обрабатывающим 8-разрядные данные и передающим их через буфер на внешнюю шину данных. Кроме того, в состав 8080 входит устройство управления, буфер адресной шины, связанный с регистром команд и блок регистров. В общих чертах работа микропроцессора выглядит следующим образом: в регистре, называемом программным счетчиком, хранится адрес следующей команды, которую необходимо выполнить. Устройство управления подключает этот регистр к шине адреса (конечно, через буфер) и выдает управляющие сигналы, необходимые для чтения кода команды из памяти. На этом завершается первый такт.

В следующем такте микропроцессор проверяет состояние сигналов на входе готовности и запрос останова. При их наличии микропроцессор переходит в соответствующее состояние. В противном случае, после появления на шине управления сигналов, подтверждающих выдачу кода команды на шину данных, устройство управления подключает к ней регистр команды и записывает в нее полученный код. Это требуется потому, что команда передается только в первом машинном цикле, а сохранить ее нужно на все время выполнения команды. Из регистра команды ее код поступает в дешифратор команды и затем в устройство управления, которое в зависимости от поступившей команды либо сразу переходит к ее выполнению, либо считывает данные или адрес, расположенные сразу после кода команды и необходимые для ее выполнения. На это тратится третий такт и, если это необходимо, четвертый и пятый такты. Таким образом, вся команда выполняется за 3-5 тактов. При тактовой частоте 2 МГц это составляет 1.5-2.5 мкс.

Перед выполнением команды проверяется состояние сигнала на входе захвата шины HLD (этот сигнал отключает микропроцессор от шины, давая внешним ус-

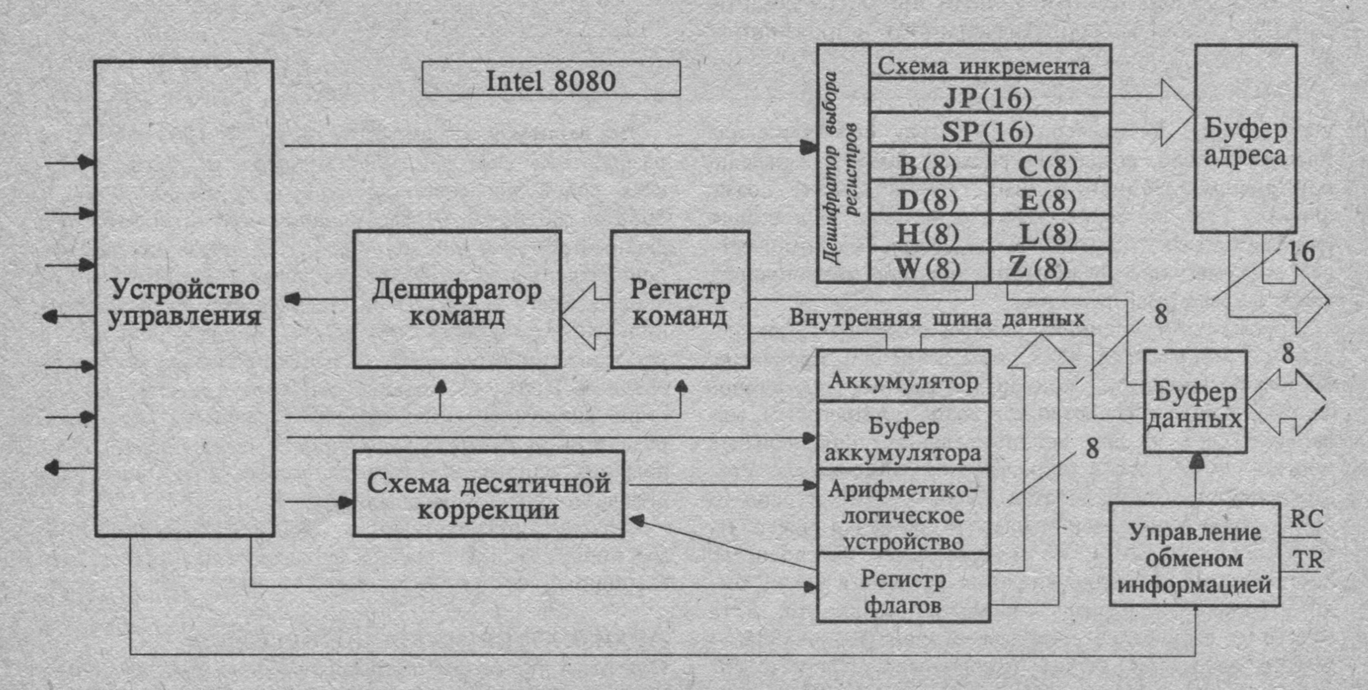


Рис.1. Внутренняя структура микропроцессора 8080.

тройствам возможность прямого доступа в память по общей шине). При его обнаружении микропроцессор переходит в состояние захвата и вырабатывает сигнал подтверждения захвата шины. После снятия сигнала HLD процессор продолжает выполнение команды. В конце машинного цикла вновь анализируется состояние входа захвата, а затем проверяется, завершено ли выполнение команды. Если нет, то микропроцессор переходит к выполнению следующего цикла команды. Это может быть цикл работы с оперативной памятью или с внешним устройством.

После выполнения каждой команды проверяется состояние входа запроса прерывания. Если этот сигнал присутствует, то текущая программа приостанавливается и на шину данных выдается сигнал "подтверждение прерывания". Затем внешний контроллер прерываний передает по шине данных команду и адрес перехода к подпрограмме обработки прерывания. По окончании обработки прерывания происходит возврат к выполнению прерванной программы.

Сигнал готовности позволяет синхронизировать работу микропроцессора с работой более медленных устройств, и, кроме того, используется для пошагового выполнения программ.

Следующим важным узлом микропроцессора является блок регистров. Он включает в себя 16-разрядный регистр для временного хранения данных WZ, шесть 8-разрядных регистров общего назначения B, C, D, E, H, L, которые могут использоваться парами в качестве 16-разрядных — BC, DE, HL (это сделано прежде всего для удобной работы с адресами). Кроме того, блок регистров содержит 16-разрядный регистр адреса команды IP (программный счетчик), 16-разрядный регистр указателя стека SP, а также 16-разрядную схему инкремента-декремента. С помощью последней изменяется, например, состояние программного счетчика после выполнения каждой следующей команды.

Еще один важный узел — регистр результата (аккумулятор), связанный с АЛУ и используемый для хранения одного из исходных операндов или результата выполнения команды.

Последний регистр — это регистр флажков. В нем записан байт, каждый бит которого содержит информацию о результате выполнения последней команды.

АРХИТЕКТУРА МИКРОПРОЦЕССОРА i8086

Микропроцессор Intel 8086 — существенно отличается от i8080. В нем применена новая значительно более мощная и гибкая система команд, есть возможность адресации 1 Мбайта памяти, обращения к 65536 устройствам ввода и такому же количеству устройств вывода информации.

В i8086 имеется возможность изменения внутренней аппаратной конфигурации с помощью специального управляющего сигнала. В более простом режиме 8086 ориентирован на использование в простых вычислительных и управляющих устройствах. При этом микропроцессор сам вырабатывает сигналы управления шиной и обеспечивает прямой доступ к ней посредством контроллера Intel 8257. В режиме полной конфигурации обеспечивается работа с контроллером шины 8288, который декодирует три сигнала состояния процессора и в зависимости от них выдает семь сигналов управления шиной. Такой режим используется в мультипроцессорных системах и в сложных вычислительных устройствах, в частности, в компьютере IBM PC/XT.

Интересно организована память: хранение 16-разрядных слов осуществляется в виде отдельных байтов, причем байты, передающиеся по восьми младшим линиям шины данных (D7-D0), собраны в банк 0, а передаваемые по восьми старшим линиям — в банк 1. Объем каждого банка составляет 512 Кбайт. Таким образом, нечетные байты хранятся в банке 1, а четные — в банке 0. Выбор банка осуществляется с помощью младшего адреса и сигнала управления старшими разрядами шины данных.

Еще одна важная особенность — возможность обработки 256 типов прерываний (от 0 до 255), в том числе есть прерывания, определяемые пользователем, и пошаговые прерывания.

Микропроцессор Intel 8086 приспособлен для работы с несколькими процессорами в одной системе, причем возможно использование как независимых процессоров, так и сопроцессоров. Отличие заключается в том, что независимый процессор выполняет свою собственную последовательность команд, а сопроцессор следит за потоком команд центрального процессора и выделяет из него "свои" команды, расширяя набор команд основного процессора и улучшая таким образом характеристики системы. Для поддержки этих режимов используются команды ESC, LOCK и XCHG, а также специальные управляющие сигналы, позволяющие разрешать конфликты доступа к общим ресурсам.

Внешние шины адреса и данных в 8086 объединены, и поэтому наличие на шине в данный момент времени информации или адреса определяется порядковым номером такта внутри цикла. Процессор ориентирован на параллельное выполнение команды и выборки следующей команды. В целом выполнение команды происходит примерно так же, как и в 8080. Команда выбирается из памяти и принимается микропроцессором в свободный регистр очереди команд, причем в то же самое время выполняется предыдущая команда. Конвейеризация команд позволяет значительно повысить быстродействие системы. При выполнении команд проверяются состояния входов запросов прерываний и захвата шины, и при необходимости выполняются соответствующие действия.

Микропроцессор i8086 состоит из трех основных частей: устройства сопряжения шины, устройства обработки и устройства управления и синхронизации.

Устройство сопряжения шины состоит из шести 8-разрядных регистров очереди команд, четырех

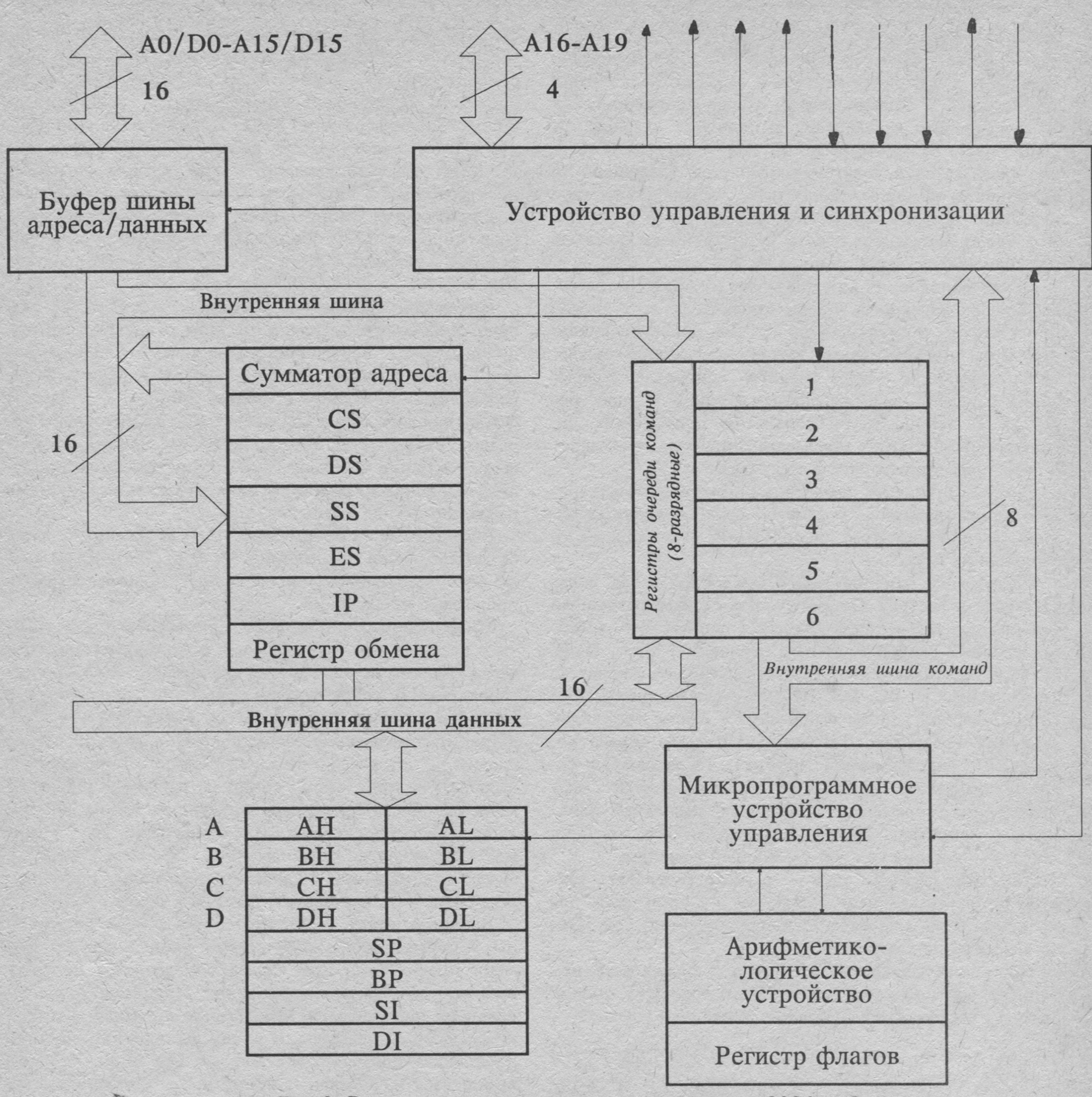


Рис.2. Внутренняя структура микропроцессора 8086.

16-разрядных регистров адреса команды, 16-разрядного регистра команды и 16-разрядного сумматора адреса (см. рис. 2). Оно выполняет следующие функции: выбирает команды из памяти и записывает их в регистр очереди команд, вычисляет и формирует физический адрес, читает операнды из памяти или из регистров и записывает результат выполнения команд в память или в регистры.

Устройство обработки преобразует данные. Команда из очереди команд по запросу устройства обработки поступает на внутреннюю шину команд, а с нее на микропрограммное устройство управления, декодирующее ее и генерирующее соответствующие последовательности микрокоманд, необходимые для выполнения текущей операции. В отличие от первых микропроцессоров, устройство обработки в 8086 не связано с внешней шиной, а обменивается с ней информаций через регистр обмена устройства сопряжения шины.

Устройство обработки содержит 16-разрядное арифметико-логическое устройство, восемь 16-разрядных

регистров общего назначения и 16-разрядный регистр флагов. Регистры могут использоваться как 16-разрядные или как пары 8-разрядных (при этом их количество удваивается).

АРХИТЕКТУРА МИКРОПРОЦЕССОРА i80286

Кристалл 80286 представляет для читателя интерес прежде всего потому, что является, пожалуй, наиболее распространенным микропроцессором из применяющихся в персональных компьютерах. Как и его предшественник — 8086 — он имеет 16-разрядные шины данных и адреса и самым характерным его отличием можно считать, помимо большей тактовой частоты, возможность работы в режиме виртуальной адресации (адресация памяти объемом более 1 Мбайта), речь о котором пойдет ниже.

Регистры

Как и любой процессор, Intel 80286 содержит некоторое количество ячеек памяти быстрого доступа, называемых регистрами. В состав i286 входят три набора по четыре регистра и один специальный регистр — указатель команды.

Регистры общего назначения

Первый набор включает в себя регистры общего назначения или РОН, необходимые для временного хранения тех операндов и результатов вычислений, доступ к которым постоянно повторяется в процессе выполнения программы. Использование РОН в подобных случаях существенно ускоряет работу системы за счет сокращения времени чтения/записи и пересылки данных из ОЗУ. Всего регистров общего назначения четыре, они, разумеется, 16-разрядные, но могут использоваться и как 8-разрядные (однобайтные), при этом их количество удваивается.

	7 0	7 0	
AX	AH	AL	Аккумулятор
BX	BH	BL	База
CX	CH	CL	Счетчик
DX	DH	DL	Данные

Рис.3 Регистры общего назначения

Функции всех РОН, в основном, идентичны, но в некоторых случаях архитектура предполагает их строгую специализацию. Например, при выполнении команд обработки строк и циклов, в одном из регистров должно храниться число, равное количеству итераций. Этот регистр выполняет роль счетчика (counter) и носит название CX. Остальные регистры выполняют функции аккумулятора (AX), базы (BX) и ячейки временного хранения данных (DX). Как мы уже знаем, каждый регистр из числа РОН может быть разделен на два однобайтных, один из которых (0-7) называется младшим (Low), а другой (7-15) — старшим (High). В соответствии с этим, каждый 8-разрядный регистр получил свое название: младшие именуются AL, BL, CL, DL, а старшие — AH, BH, CH и DH (рис. 3).

Перед тем как познакомиться с назначением и функциями остальных наборов регистров, разберемся, каким образом процессору с 16-разрядной шиной адреса удается работать с памятью объемом в 1 Мбайт.

Режим реального адреса

Адресная шина процессора 80286 имеет ширину 16 бит, к тому же известно, что максимальное двоичное число длиной в два байта равно 2^{16} или 64 Кбайт и, если адрес задается таким числом, то, вроде бы, пространство ОЗУ, с которым может работать процессор, не должно превышать 64 Кбайт. С другой стороны, 1 Мбайт памяти можно адресовать с помощью двоичного числа длиной 20 бит (2^{20}). Как быть?

Можно, например, воспользоваться двойным адресом, ведь в повседневной жизни нам приходится постоянно сталкиваться с многоступенчатыми адресами: мы пишем на почтовом конверте сначала название города, потом — улицы, дома и т.д. (Предположим на мгновенье, что все квартиры в СССР пронумерованы последовательно, каково придется почте в такой ситуации?) Разработчики 80286 решили проблему подобным же образом: полный адрес ячейки памяти состоит из комбинации двух 16-разрядных чисел, причем одно из них предназначили для адресации внутри некоторой области ОЗУ размером 64 Кбайта, а второе — для локализации этой области во всем пространстве ОЗУ. Область, внутри которой происходит адресация, называется сегментом, а адрес внутри сегмента — внутрисегментным смещением. Адрес, локализующий положение сегмента в оперативной памяти, содержится в одном из специальных сегментных регистров процессора, но он тоже 16-разрядный. Для того, чтобы при помощи этого адреса можно было перекрыть все пространство ОЗУ, со стороны младшего байта его дополняют четырьмя нулями. Например, если содержимое сегментного регистра: 0001.1101.1000.1111 (или $1D8F_{16}$) то адрес начала соответствующего сегмента будет равен: 0001.1101.1000.1111.0000 (или $1D8F0_{16}$). Таким образом можно искусственно разделить всю память на сегменты, начинающиеся по адресам, кратным 16_{10}. Предположим, что внутрисегментное смещение нашей ячейки задано числом 1001.1011.0010.0101 или $9B25_{16}$, в этом случае ее реальный адрес будет равен сумме адреса сегмента и внутрисегментного смещения: $1D8F0_{16}+9B25_{16} = 27015_{16}$ (рис. 4).

Выполняемая программа может обращаться к любому из четырех сегментов, именуемых: текущий сег-

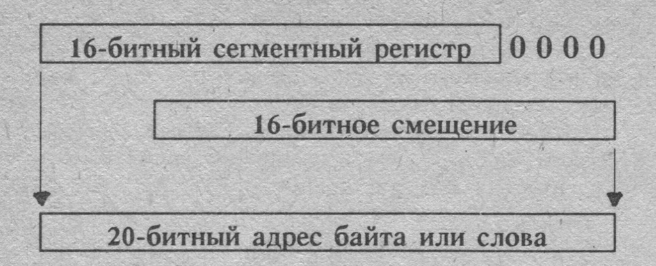


Рис.4 Формирование адресов байта или слова

мент кода (то есть программы), текущий сегмент данных, текущий сегмент стека и текущий дополнительный сегмент.

Теперь вернемся к регистрам.

Указательные и индексные регистры

Второй набор состоит из четырех 16-разрядных регистров, которые, в основном, используются для хранения внутрисегментных смещений. При выполнении многих команд функции каждого из данных регистров строго определены.

Регистры SI (индекс источника) и DI (индекс приемника) называются указательными и содержат смещения в текущем сегменте данных. Регистры SP (указатель стека) и BP (указатель базы) называются базовыми и содержат смещения в текущем сегменте стека (рис. *5*). Для тех, кто забыл или не знает, напомню, что стек — это способ организации работы с оперативной памятью по принципу LIFO (Last In — First Out), что в переводе означает: "последним вошел — первым вышел", то есть слово данных, помещенное в стек последним, будет извлечено оттуда в первую очередь. Само слово стек произошло от английского stack — скирда. И действительно, область памяти, организованную в виде стека, можно сравнить со скирдой сена: последние, уложенные сверху снопы будут использованы первыми — никому не придет в голову выгребать сено из середины. Для работы со стеком необходимо знать две величины: адрес дна и адрес вершины стека. Если адрес дна — число фиксированное, то адрес вершины зависит от того, сколько байтов данных содержится в стеке. В нашем случае адрес вершины находится в регистре SP.

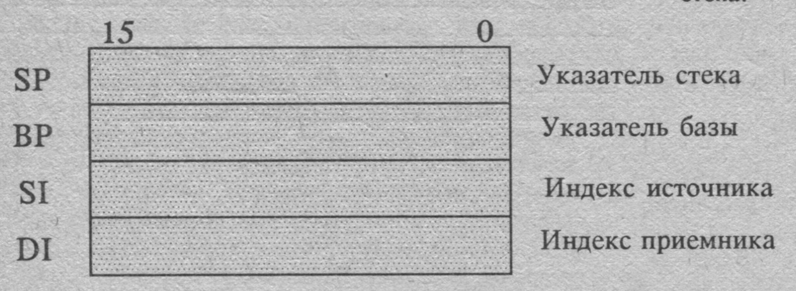


Рис.5 Указательные и индексные регистры

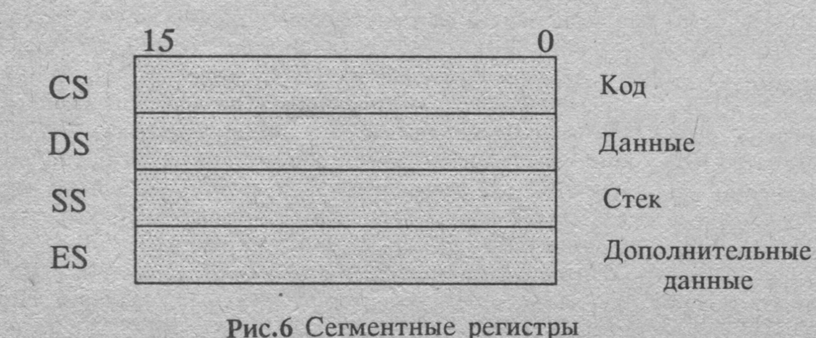


Рис.6 Сегментные регистры

Регистры SI и DI содержат смещения, соответственно, источника и приемника при выполнении команд обработки строк MOVSB, MOVSW, LODSB, LODSW, STOSB, STOSW и команды LOOP.

Сегментные регистры

Если мы вспомним, что программа в любой момент может обратиться к одному из четырех сегментов: к текущему сегменту кода, данных, стека или к дополнительному (сегменту данных), то нас вряд ли удивит, что в состав процессора входят четыре 16-разрядных регистра, являющихся указателями адресов текущих сегментов. Их функции строго дифференцированы, а потому каждый регистр имеет свою "профессию": CS определяет сегмент кода, DS — сегмент данных, SS — сегмент стека и ES — дополнительный сегмент (рис. 6).

Теперь для того, чтобы, к примеру, произвести выборку слова данных из стека, программе достаточно обратиться к регистрам SS и SP, сложить находящиеся в них числа по уже известному нам правилу и в качестве результата получить реальный адрес вершины стека.

Флажки

Не знаю как для вас, а для нас более привычно звучит термин "слово состояния", ведь, собственно говоря, совокупность значений флажков и определяет состояние процессора во время его работы. В самом общем случае слово состояния — это двоичное число, каждый бит которого отражает строго определенный параметр состояния устройства. Что касается 80286, то здесь биты слова состояния называются флажками, всего их девять, причем шесть из них регистрируют состояние процессора, а три — применяются для управления его работой (рис. 7).

К флажкам состояния относятся:
флажок переноса CF (имеет значение равное 1 при переносе из старшего бита)
флажок вспомогательного переноса AF (индицирует перенос из младших 4-х бит)
флажок переполнения OF (устанавливается равным единице при выходе знакового результата за границу диапазона)

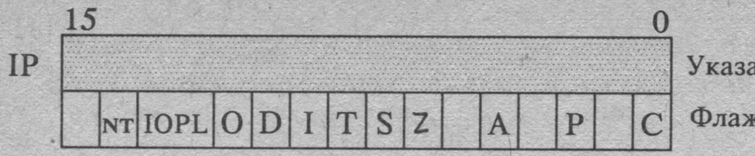


Рис.7 Указатель команды и флажки

флажок нуля ZF (фиксирует нулевой результат выполнения команды)
флажок знака SF (фиксирует отрицательный результат выполнения команды)
флажок четности PF (фиксирует четное число единиц в последнем байте, полученном в результате выполнения команды)

К флажкам управления относятся:
флажок направления DF (указывает направление прохождения строк в строковых командах)
флажок разрешения прерывания IF (разрешает или запрещает прерывание по входу INTR)
флажок трассировки TF (переводит процессор в пошаговый режим)

В следующем номере мы расскажем вам о режимах адресации процессора i286.

И.Вязаничев, И.Липкин

Обучающий курс журнала LAN Magazine представляет собой серию статей по вопросам локальных сетей для начинающих пользователей. В этом курсе в простой и доступной форме излагаются основные концепции, лежащие в основе организации локальных сетей. Каждый месяц в сборнике КомпьютерПресс будет печататься очередной выпуск серии, посвященный какому-либо вопросу, связанному с организацией локальных сетей. Вырезайте и сохраняйте выпуски серии и вы сможете получить в конце обучающего курса брошюру, которая будет представлять собой введение в локальные сети. В этом выпуске будут рассматриваться вопросы, связанные с сетевыми возможностями операционной системы OS/2

Локальные сети от А до Я:

курс обучения

ЧАСТЬ 18. СЕТЕВЫЕ ВОЗМОЖНОСТИ ОПЕРАЦИОННОЙ СИСТЕМЫ OS/2

Как уже не раз отмечалось в нашем обучающем курсе, основная причина появления и бурного развития локальных сетей была связана с необходимостью обеспечения программной и аппаратной поддержки коллективного использования дорогостоящих периферийных устройств (принтеров, жестких дисков и т.п.). Однако, по мере развития вычислительной техники и, прежде всего, в связи со значительным увеличением мощности персональных компьютеров, которая стала сравнима с мощностью мини- и универсальных ЭВМ, пользователи локальных сетей получили новые уникальные возможности. Так, персональные компьютеры с быстродействием в несколько миллионов и даже десятков миллионов операций в секунду, появившиеся в последнее время на рынке, позволяют поддерживать многозадачный режим работы рабочих станций и файл-серверов, что значительно повышает эффективность использования локальной сети. Именно возможность использования многозадачного режима работы было, до последнего времени, привилегией мини- и универсальных ЭВМ, где применялись соответствующие операционные системы типа UNIX или VMS. Поскольку, в связи с появлением микропроцессоров Intel 80286 и 80386, аппаратные средства персональных компьютеров уже не сдерживали продвижения вперед к многозадачному режиму работы, разработка первой многозадачной операционной системы не заставила себя долго ждать. Ею стала OS/2 — детище фирм IBM и Microsoft.

Многозадачный режим работы OS/2

Как известно, персональные компьютеры, в которых используется один центральный процессор, не могут выполнять параллельно несколько вычислений. Однако, быстродействие современных микропроцессоров позволяет выполнять вычисления с такой скоростью, что часто создается впечатление параллельной обработки нескольких заданий. Для полной загрузки современных микропроцессоров уже сейчас требуются задачи с большими объемами расчетов, что не всегда реально для одного пользователя. Это приводит к тому, что микропроцессоры часто попросту простаивают, т.е. находятся в состоянии ожидания. С целью сокращения времени ожидания и используется многозадачный режим.

Предположим, что вы работаете с каким-либо текстовым редактором, например Microsoft Word, и печатаете статью о принципах организации многозадачного режима в персональных компьютерах. При нажа-

тии клавиш на клавиатуре компьютер фиксирует буквы и выводит их на экран. Все эти действия выполняются микропроцессором настолько быстро, что большую часть времени между нажатиями клавиш он находится в состоянии ожидания. Именно эти паузы и использует операционная система OS/2 для организации многозадачного режима работы персонального компьютера. В то время, когда вы печатаете вашу статью, OS/2 может загрузить в память файл из другого компьютера, пересчитать таблицу, отсортировать базу данных или начертить график. Для одновременного выполнения всех указанных операций в OS/2 используется принцип квантования времени (slicing), который заключается в том, что все время работы микропроцессора делится на небольшие отрезки времени — кванты. Обычно продолжительность кванта времени составляет долю секунды, например, в микропроцессоре 80286 он равен 32 мс. При многозадачном режиме работы на выполнение каждого задания компьютер отводит один или несколько последовательных квантов времени, после чего переходит к выполнению другого задания. Так, например, на ввод и обработку буквы при работе с текстовым редактором компьютеру требуется один квант времени. В течение следующего кванта компьютер может выполнить пересчет таблицы, а затем сортировку базы данных. Таким образом, компьютер переходит от одной задачи к другой до тех пор, пока все они не будут полностью завершены. Поскольку кванты времени достаточно малы, то работая, скажем, над статьей в редакторе Microsoft Word, вы практически не будете замечать, что ваш компьютер выполняет еще несколько заданий.

Виртуальность OS/2

Для организации многозадачного режима работы персонального компьютера в OS/2 используются виртуальные устройства и виртуальная память. Принципы организации виртуальной памяти и работы с виртуальными устройствами давно применяются в мини- и универсальных ЭВМ, но лишь с появлением микропроцессоров 80286 и 80386 стало возможно их использование в персональных компьютерах. Это связано с тем, что микропроцессоры 80286 и 80386 могут работать в так называемом защищенном режиме (protected mode).

Чтобы понять суть работы микропроцессора 80286 или 80386 в защищенном режиме, следует сделать шаг назад и вспомнить, что более ранняя модель фирмы Intel — микропроцессор 8086/8088 поддерживает лишь реальный режим (real mode). При работе в реальном режиме программы пользователя имеют прямой доступ ко всем внешним устройствам персонального компьютера, включая память, клавиатуру, экран и порты ввода/вывода. Это приводит к тому, что программы могут выполнять любые операции с внешними устройствами без участия операционной системы. Так, например, текстовый редактор может "принять решение" о выдаче данных на печатающее устройство в обход DOS. Если в это время какая-либо другая программа уже работает с принтером, то может возникнуть конфликт, предотвратить который в рамках реального режима работы невозможно, а значит и невозможно организовать многозадачную среду. Самое неприятное происходит тогда, когда программы начинают перезаписывать друг друга в памяти компьютера. Поскольку в реальном режиме работы ничто не может помешать одной программе, скажем, электронной таблице, использовать ту же область памяти, которую уже использует какая-либо СУБД для сортировки базы данных, то возникает ситуация, когда вторая программа (в нашем случае электронная таблица) не только портит данные первой программы (СУБД), но и может перезаписать часть самого кода первой программы. Это, как правило, приводит к "замораживанию" компьютера, поскольку последний не может продолжить выполнение первой программы.

При работе в защищенном режиме программы, как правило, не имеют прямого доступа к внешним устройствам персонального компьютера, то есть внешние устройства как бы защищены от захвата прикладными программами. Отсюда и название — защищенный режим. Так как у программ нет прямого доступа к внешним устройствам, то им необходим какой-либо "посредник" для связи с ними. В качестве такого "посредника" и выступает операционная система OS/2. Она как бы располагается между программами и внешними устройствами, регулируя работу с последними и обеспечивая базу для работы в многозадачном режиме. Для этого в OS/2 используются так называемые виртуальные устройства (virtual devices) и виртуальная память (virtual memory), которая, по существу, также представляет собой особое виртуальное устройство.

Прежде всего рассмотрим работу виртуальной памяти. В отличие от операционной системы DOS, где программы обращаются к оперативной памяти лишь по физическим адресам, в OS/2 вводится дополнительный шаг — вместо запроса физического адреса памяти программы запрашивают адрес виртуальной памяти, а OS/2 преобразует этот виртуальный адрес в физический. Именно с помощью такого преобразования OS/2 предотвращает перекрытие адресных пространств различных программ, поскольку виртуальные адреса отображаются лишь в свободные, не занятые другими программами, физические адреса.

Кроме того, OS/2 может обеспечить отображение виртуального адреса на жесткий диск. При этом появляется возможность разместить данные и саму программу частично на диске и подгружать их по мере необходимости. Сама OS/2 принимает решение о том, где лучше расположить часть программного кода или данных: на жестком диске или в оперативной памяти. Для организации обмена данными между памятью и диском OS/2 делит адресное пространство компьютера на блоки по 64 Кбайт, которые называются сегментами. Если в оперативной памяти нет места для полной загрузки какой-либо программы, то сегменты другой, уже загруженной программы могут быть записаны на

диск, чтобы освободить место для новой программы. По мере выполнения каждой программы OS/2 выполняет автоматическую подкачку соответствующих данных с жесткого диска. Благодаря использованию виртуальных адресов и автоматической подкачке сегментов, OS/2 предоставляет пользователям практически неограниченные ресурсы памяти — до 1 Гбайта.

Подобным образом в OS/2 организуется работа с другими виртуальными устройствами. Вместо прямого доступа к внешним устройствам, таким как порты ввода/вывода или экраны, программам предоставляются виртуальные экраны и порты. При этом каждая программа как бы имеет свою клавиатуру, экран, принтер, модем и т.п., которые OS/2 отображает в реальные физические устройства.

Возможность работы в OS/2 с виртуальными устройствами и виртуальной памятью позволяют выполнять на персональном компьютере несколько программ одновременно, т.е. поддерживать многозадачный режим. Именно многозадачный режим работы компьютера является основой для организации многопользовательской среды, которая значительно улучшает сетевые характеристики OS/2.

Многопользовательская среда OS/2

Основной задачей многопользовательской среды является прием запросов от различных пользователей и направление результатов уже выполненных запросов к своим адресатам. Как известно, сами по себе ни DOS, ни OS/2 не поддерживают такую переадресацию данных — для этого необходимо использовать какую-либо сетевую операционную систему. Различие сетевых характеристик DOS и OS/2 заключается лишь в уровне сервисных функций, которые они могут предоставить сетевой операционной системе.

Сначала вспомним, каким образом работает сетевая операционная система в среде DOS. По мере поступления каждого запроса пользователя, сетевая операционная система передает его в DOS на выполнение и контролирует возврат ответов на запросы соответствующим адресатам. Однако DOS имеет определенные ограничения на обработку сетевых запросов. Так, DOS может выполнять только самые простые операции ввода/вывода, обслуживающие пользователей сети, такие, как физическое считывание и запись с диска и на диск или вывод данных на принтер. Таким образом, сеть персональных компьютеров, работающая на базе DOS, может поддерживать только коллективное использование периферийных устройств (дисков, принтеров и т.п.).

Иная картина складывается при работе с OS/2. Здесь имеется такой мощный инструмент, как многозадачный режим работы. С его помощью сетевая операционная система на базе OS/2, например, LAN Manager фирм 3Com и Microsoft или LAN Server фирмы IBM, может выполнять переадресацию различных виртуальных устройств OS/2, что позволяет организовать обработку заданий нескольких пользователей сети на одном персональном компьютере и, таким образом, коллективно использовать его микропроцессор. Например, в течение одного из квантов времени OS/2 выполняет построение графика для удаленного пользователя, а сетевая операционная система переадресует вывод с виртуального экрана OS/2 по сети на физический экран удаленного пользователя. Затем, в следующий квант времени OS/2 воспринимает сигналы с клавиатуры другого пользователя, который может находиться на другом конце сети, и отображает эти клавиши на виртуальную клавиатуру. Таким образом, виртуальные устройства и многозадачность помогают OS/2 создать базу для работы в многопользовательском режиме. Это качественно отличает сетевые характеристики OS/2 и DOS, поскольку в OS/2 имеется возможность не только коллективно эксплуатировать периферийные устройства, но и работать в многопользовательском режиме, что очень важно для организации локальных сетей.

Еще одной существенной особенностью OS/2 является обеспечение связи между выполняемыми задачами (процессами). Но об этом — в следующем выпуске обучающего курса.

В.Миропольский, В.Демидов

По материалам:
"LAN tutorial series", LAN Magazine, September 1989.

Фирмы IBM и 3Com выпустили проект совместно разработанной спецификации локальных сетей. Она должна стимулировать действия различных фирм в направлении усовершенствования взаимодействия разнородных систем.

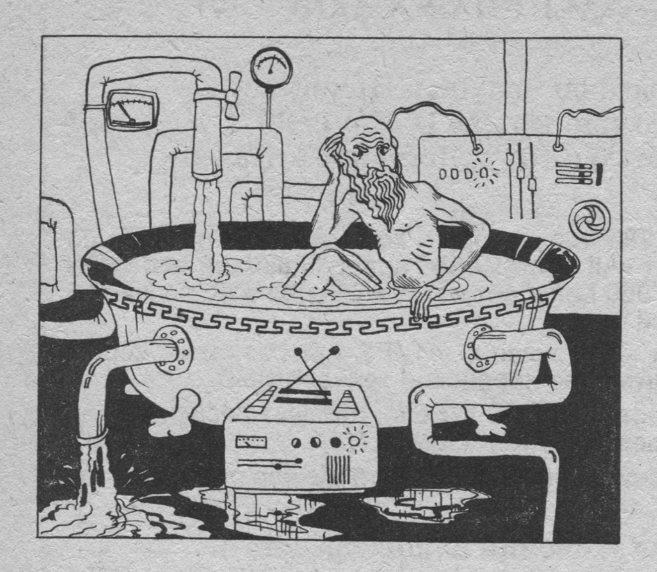
Проект Heterogeneous LAN Management обеспечивает создание инфраструктуры для разработки продуктов, функционирующих в среде сетевых операционных систем.

Предлагается поддержка нескольких типов интерфейсов между операционной системой и сетевыми адаптерами независимо от вида кабелей, объединяющих их в сеть. "Возможность совместной работы является ключом к современной вычислительной среде, где к оборудованию различных производителей предъявляются требования совместного функционирования и эффективного управления", — заявил В.Месснер, один из руководителей фирмы 3Com.

Перед публикацией проект спецификации был рассмотрен техническими группами четырех ведущих фирм-производителей программ — Banyan, Microsoft, Novell и Santa Cruz Operation. В начале декабря комитет-802 Института инженеров по электронике и электротехнике (IEEE) проголосовал в поддержку разработанного подхода.

Как IBM, так и 3Com заявили об участии в разработке стандарта IEEE 802.1B для управления сетями. Этот стандарт позволит разработчикам сетевых программ использовать минимум памяти для поддержки спецификаций Token Ring (IEEE 802.5) и Ethernet (IEEE 802.3). Проект спецификации HLM можно получить бесплатно в фирмах IBM или 3Com.

Newsbytes News Network, December 21, 1990



АВТОМАТИЗАЦИЯ НАУЧНЫХ ИССЛЕДОВАНИЙ

В научной деятельности одно из первых мест занимают приборы и инструменты. Поэтому в настоящее время для повышения эффективности научных исследований важное значение приобретает автоматизация научных исследований, позволяющая не только автоматизировать эксперимент, но и осуществлять моделирование исследуемых объектов, явлений и процессов, изучение которых традиционными средствами затруднено или невозможно. Решению этой задачи призваны служить автоматизированные системы научных исследований (АСНИ).

Персональные компьютеры в АСНИ могут использоваться для решения четырех основных задач. Во-первых, ПК могут успешно взять на себя управление экспериментом. Во-вторых, ПК могут быть эффективны при подготовке отчетов и документации как компоненты встроенной конторской подсистемы. В-третьих, во многих приложениях на ПК можно возложить поддержание базы экспериментальных данных (в настоящее время имеются персональные автономные накопители большой емкости — до 800—1400 Мбайт, а там, где вычислительных мощностей ПК недостаточно, они могут входить в состав локальной вычислительной сети как рабочие станции, обеспечивающие доступ к центральной БД). Наконец, в-четвертых, ПК могут выступать в качестве технического средства при построении информационно-поисковых, библиографических и экспертных систем. В зависимости от конкретного приложения соотношение указанных компонентов может быть различным, например, для управления рядом физических экспериментов, отличающихся высокой сложностью и спецификой процессов, может потребоваться специализированное оборудование, высокопроизводительные ЭВМ и т.п.

Эффективность применения ПК в автоматизации научных исследований заключается в следующем. Во-первых, в несколько раз сокращается цикл исследования (экспериментирования) за счет ускорения подготовки и проведения эксперимента, оперативного использования результатов экспресс-анализа, проводимого в реальном масштабе времени, сокращения времени обработки и систематизации данных, уменьшения числа ошибок при измерении и обработке. Во-вторых, увеличивается точность результатов и их достоверность, так как в АСНИ возможно использование методов, снижающих влияние накапливающихся ошибок округления при вычислении промежуточных результатов. В-третьих, повышается качество и информативность эксперимента за счет увеличения числа контролируемых параметров (по сравнению с "некомпьютерными" исследованиями) и более тщательной обработки данных. В-четвертых, в ходе интерактивного взаимодействия с АСНИ достигаются усиление контроля за ходом эксперимента и возможность его оптимизации. В-пятых, сокращается штат участников эксперимента, повышается производительность исследователя. Наконец, очень важным является то, что результаты экспериментов структурируются и выводятся оперативно в наиболее удобной форме — графической или символьной. Например, вместо того, чтобы просматривать километровые таблицы данных, их можно компактно структурировать в виде графических объектов. Так, зависимость от двух аргументов очень удобно представлять средствами трехмерной графики в виде "горных массивов" — в последние можно интегрировать многие миллионы измерений, что невозможно в обычной табличной форме. Наиболее интересные участки "горных массивов" можно увеличить, получив детальное представление о поведении функции и т.п. Выигрыш достигается колоссальный.

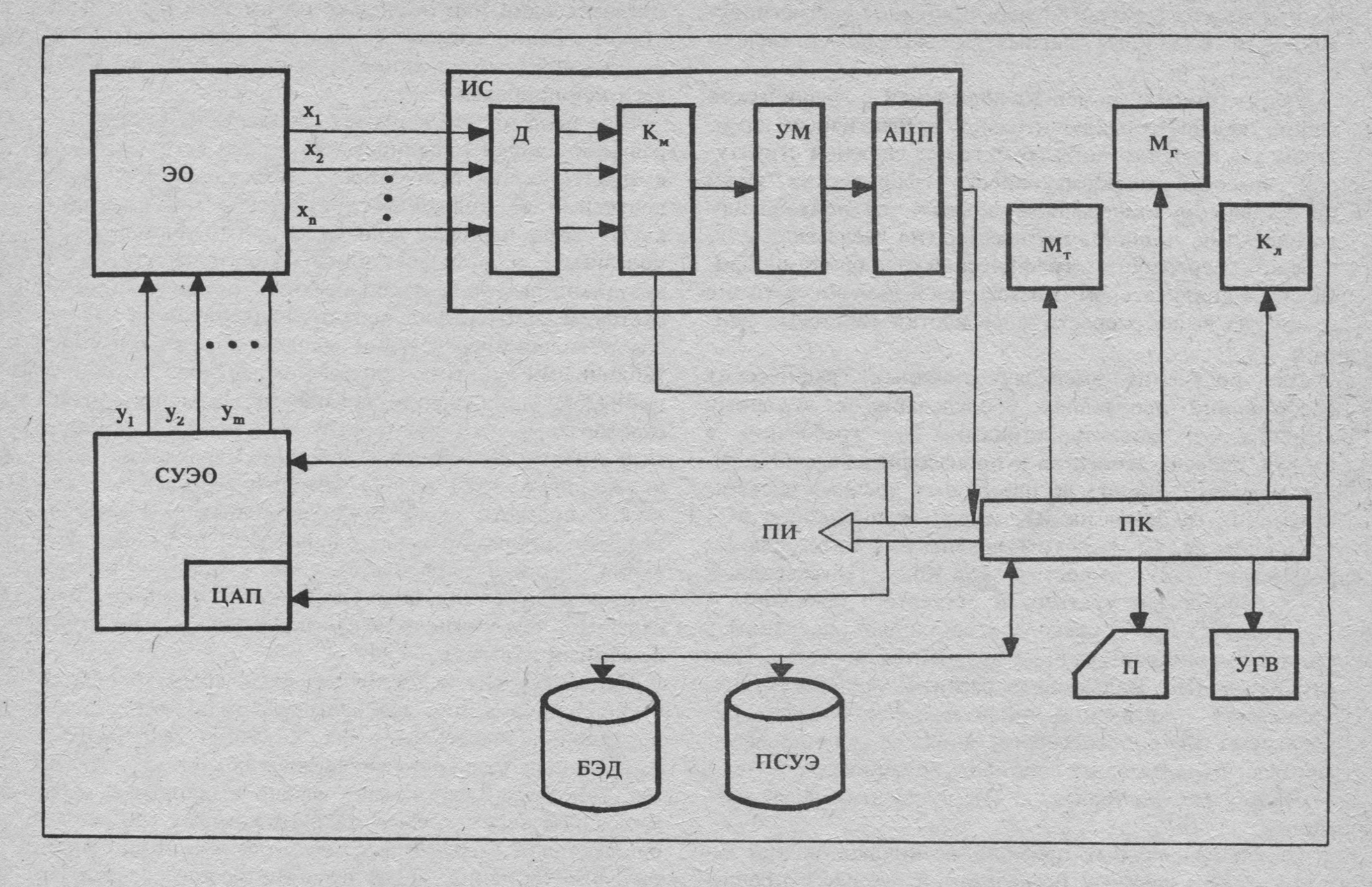


Рис. Типовая схема автоматизации экспериментальных исследований:

ЭО	— экспериментальный объект;
$x_1, x_2, ..., x_n$	— измеряемые параметры;
ИС	— измерительная система;
Д	— датчики;
$К_м$	— коммутатор;
УМ	— усилитель-масштабатор;
АЦП	— аналого-цифровой преобразователь;
ПИ	— приборный интерфейс;
ЦАП	— цифро-аналоговый преобразователь;
СУЭО	— подсистема управления экспериментальным объектом;
$y_1, y_2, ..., y_m$	— управляющие сигналы;
ПК	— персональный компьютер;
$М_т$	— монитор текстовый;
$М_г$	— монитор графический;
$К_л$	— клавиатура;
УГВ	— устройство графического вывода (графопостроитель);
П	— печатающее устройство;
БЭД	— база экспериментальных данных;
ПСУЭ	— программная система управления экспериментом

На рис. показана общая схема подключения ПК в контур управления экспериментом. В конкретных реализациях могут быть незначительные отклонения от приведенной схемы. Например, в конкретной реализации может предусматриваться только цифровое управление, тогда не нужен цифро-аналоговый преобразователь или, наоборот, необходим только аналоговый (в случае отсутствия связи от приборного интерфейса непосредственно к системе управления экспериментальным объектом). В качестве устройства вывода графической и текстовой информации используется один графический дисплей и один матричный принтер и т.п.

Современные ПК, обладая высокими техническими характеристиками, позволяют использовать их в таких необычных на первый взгляд приложениях, как измерительные приборы, различного рода регистраторы и осциллографы, путем простого программирования и подключения соответствующих дополнительных устройств. На экране одного графического дисплея возможно формирование целой системы приборных шкал (вольтметров, амперметров, омметров, фотометров и

многих других измерительных приборов), регистрирующих те или иные параметры экспериментального объекта.

Таким образом, обмен информацией в графической форме является исключительно эффективным средством для представления объектов со сложной структурой. Высокая информативность графических форм представления информации объясняется психофизиологическими свойствами восприятия человека: так, скорость переработки графической информации зрительным анализатором оказывается в десятки, а то и в сотни раз выше скорости переработки текстовых данных.

Для получения высококачественных графических изображений необходимо оборудование с высокими техническими характеристиками. Это требование в первую очередь относится к производительности и емкости памяти. Вплоть до настоящего времени системы машинной графики на ПК имели ограниченное распространение. С массовым появлением интегральных микросхем ОЗУ емкостью 256 Кбит, 16-разрядных микропроцессоров третьего и четвертого поколений с быстродействием в несколько миллионов операций в секунду машинная графика становится неотъемлемым атрибутом ПК. Дальнейшее развитие микроэлектронной базы — появление чипов емкостью 4 Мбит, 32-разрядных микропроцессоров, широкое использование методов параллельных вычислений приведут к еще большему распространению ПК в научных исследованиях.

Одной из сложных проблем, возникающих при автоматизации научных исследований, является проблема формы вывода многомерных данных. Если количество взаимосвязанных данных не превышает трех, то здесь принципиальных затруднений не возникает, так как возможно использование двух- или трехмерной машинной графики, например, как уже упоминалось, в виде "горных массивов". Иная ситуация возникает при попытке отображения на экране дисплея зависимостей высших размерностей. Здесь предложено много конкретных подходов, но, на наш взгляд, наиболее интересным и к тому же отвечающим "принципу дружественности" является преобразование многомерных данных в двух- или трехмерную цветоформу, легко воспринимаемую человеком. Это направление представляет огромную, фактически неразработанную область. Помимо учета характера зависимостей между многомерными данными здесь необходимо принимать во внимание и эргономические факторы. Например, в одной из интересных реализаций предложено отображать многомерные данные в виде человеческого лица, которое в зависимости от изменений данных может принимать печальное или радостное выражение. Особенно это важно при управлении сложными и опасными экспериментами.

Печальное выражение лица будет предупреждать об отклонении в ходе эксперимента от заданных параметров, и человек-экспериментатор сможет быстро вмешаться в ход эксперимента — прекратить вообще, изменить один или несколько параметров и т.п. Ясно, что изменение мимики лица воспринимается человеком намного естественнее и быстрее, чем показания десятков приборов.

Еще одно направление использования ПК связано с решением задач моделирования, часто встречающихся в практической деятельности исследователей. Здесь допустимо не только математическое моделирование какого-либо процесса или явления, традиционно используемое в исследовательской деятельности, но и визуально-натурное моделирование, которое обеспечивается за счет виртуального отображения этих процессов и явлений средствами машинной графики (а не табличными данными или графиками, как это обычно принято), т.е. перед исследователем демонстрируется своеобразный "компьютерный мультфильм", снимаемый в реальном масштабе времени. Наглядность моделирования в этом случае намного возрастает. Здесь можно провести очень близкую аналогию с компьютерными микромирами, создаваемыми для целей обучения. Разница заключается лишь в том, что в научном моделировании физические миры намного более сложные, чем учебные микромиры, которые носят упрощенный характер.

Используя ПК как универсальное средство переработки информации, экспериментатор может строить логические "заглушки" для создания виртуальных компонентов реального изучаемого объекта или системы. Например, программно можно имитировать формирование параметров, измерение которых требует дорогостоящего и громоздкого оборудования и т.п. Более того, имитационное моделирование можно распространить и на весь изучаемый объект. Рассмотрение различных имитированных вариантов позволяет исследователю выбрать оптимальный или квазиоптимальный, более надежный метод изучения явления и затем уже применить его к реальному объекту.

На высших уровнях иерархии в АСНИ находятся информационно-поисковая и экспертные системы. Первая из указанных систем предназначена для просмотра базы экспериментальных и других данных. Ограниченная емкость встраиваемой памяти на магнитных дисках (140—360 Мбайт для винчестерских накопителей) заставляет использовать автономные накопители емкостью 800—1400 Мбайт. Если и этой емкости не хватает, то ПК включаются в состав локальной вычислительной сети, управляемой мини-ЭВМ или большой ЭВМ. Однако, с ростом емкости накопителей целесообразным может стать и децентрализованное хранение информации. Большие объемы информации часто встречаются при цифровой обработке изображений, например в аэрокосмической съемке, астрофизике, ядерной физике и других подобных областях. Особое место в АСНИ отводится экспертной системе, которая, образно говоря, представляет мостик между теорией и практикой. Здесь можно проследить интересную взаимосвязь. Так, методы анализа данных, берущие свое начало в математической статистике, все усложняются и включают логические структуры, кото-

рые обеспечивают более высокий уровень обобщения информации. Это позволяет им приближаться к функциям, возлагаемым на экспертные системы, в которых имеются средства обучения открытиям. До последнего времени экспертные системы в основном создавались на больших и средних ЭВМ. Однако сейчас имеются примеры использования и ПК в экспертных системах. Постепенно с развитием баз знаний экспертные системы станут обычным атрибутом АСНИ.

Использование экспертных систем в практике исследований имеет целый ряд преимуществ. Во-первых, для решения задач и получения ответа на сложные запросы не требуется трудоемкого программирования. Если экспертная система обладает знаниями, достаточными для синтеза ответа, то ответ будет выдан. Это делает экспертные системы доступными неподготовленным пользователям, непрофессионалам в области программирования, но являющимся квалифицированными специалистами в своих предметных областях. Кроме того, "интеллектуальность" экспертных систем облегчает освоение навыков работы с ними. Во-вторых, экспертная система способна, как правило, объяснять человеку, почему и как она пришла к тому или иному результату. Это значительно повышает доверие человека к совету или решению, предложенному вычислительной машиной, и создает у него психологическую готовность использовать этот совет или решение в своей деятельности. В-третьих, экспертная система, база знаний которой построена на основании совокупных знаний группы специалистов, располагает большими интеллектуальными возможностями, чем каждый из специалистов в отдельности. По-видимому, это обусловлено синергетическим эффектом. В-четвертых, экспертная система способна к обучению — пополнению базы знаний новыми знаниями. В перспективе экспертные системы будут способны к самообучению, что еще более увеличит их возможности. Такие системы уже существуют, но они пока имеют экспериментальный, исследовательский характер.

Примеры промышленных АСНИ на базе персональных компьютеров

В настоящее время АСНИ выпускаются в виде как специализированных микрокомпьютерных систем, так и прикладных пакетов широкого назначения, что определяется стоящими целями, а также экономическими соображениями.

Рассмотрим некоторые промышленные системы АСНИ. Фирма Laboratory Technologies Corp. разработала пакет LABTECH Real Time Access, представляющий собой интерактивную систему накопления и статистической обработки данных, работающую в реальном масштабе времени. Система ориентирована на операционную систему MS DOS и персональные компьютеры фирмы IBM и обеспечивает совместимость с большинством существующих СУБД и интегрированных пакетов (dBASE, Symphony, Lotus 1-2-3 и др.). Пакет прикладных программ SYSTAT фирмы Systat один из самых мощных пакетов по математической статистике. Он включает очень широкий набор возможностей, в том числе и таких сложных, как кластер-анализ, непараметрическая статистика, анализ временных рядов, нелинейная регрессия и корреляционный анализ. Пакет предназначен для работы с машинами Apple, Macintosh и IBM PC XT/AT. Фирма STSC на рынке программного обеспечения АСНИ представлена пакетом STATGRAPHICS, отличительной особенностью которого является развитая машинная графика, допускающая возможность построения трехмерных цветных изображений. Система построена по принципу интерактивного меню и включает более 250 математико-статистических процедур.

Большой популярностью среди научных работников пользуются интегрированные пакеты АСНИ. Примером такого пакета является система ASYSTANT (разработка фирмы MacMillan Software Co.). Система имеет очень удобный человеко-машинный интерфейс, управление которым основано на полиэкранных меню. Встроенные процедуры обеспечивают выполнение следующих функций: быстрое преобразование Фурье, сглаживание кривых, интегрирование и дифференцирование аналитических выражений, подбор кривой по точкам, статистическую обработку данных, решение дифференциальных уравнений, выполнение матричных операций над алгебраическими многочленами. В расширенной версии ASYSTANT+ кроме перечисленных возможностей имеется подсистема накопления данных, обеспечивающая вывод информации в формате широко распространенных промышленных приборов — координатных и ленточных самописцев, регистраторов данных и т.п.

Базовый комплект Advanced Scientific Analysis & Graphics фирмы Simplification UnLtd. включает 47 программ, написанных на расширенной версии Бейсика для ПК IBM PC. На дистрибутивном диске имеются и тексты исходных программ, что позволяет их легко модифицировать и приспосабливать к конкретным потребностям. Все программы разбиты на следующие четыре категории:

— процедуры машинной графики, позволяющие вычерчивать трехмерные поверхности, строить каркасные фигуры и секционированные изображения;

— процедуры обработки изображений и преобразований, включающие быстрое преобразование Фурье, построение фрактальных структур, программы интегральной свертки, масштабирование изображений и программы трассировки световых лучей;

— процедуры теории вероятности и математической статистики, включающие построение вероятностных распределений, построение доверительных границ, метод моделирования Монте-Карло и вычисление регресии;

— процедуры матричной алгебры, позволяющие оперировать матрицами до третьего порядка включительно; матричные операции включают символьное умножение, вычисление определителей, транспониро-

вание, вычисление характеристических чисел матриц и векторов.

Пакет MathCAD фирмы MathSoft позволяет в интерактивном режиме создавать, редактировать и отображать на экране дисплея широкий класс функций, решать уравнения, заданные в аналитической или графической форме. В созданные графики может быть встроен любой поясняющий текст, а сами графики сохраняются в базе данных и впоследствии в любом текстовом документе. Система MathCAD имеет встроенные тригонометрические и гиперболические функции, позволяет оперировать как действительными, так и комплексными числами, использовать различные системы единиц, например международную систему СИ. Кроме того, встроенный синтаксический анализатор выполняет проверку синтаксической правильности вводимых формул. Пакет ориентирован на ПК фирмы IBM.

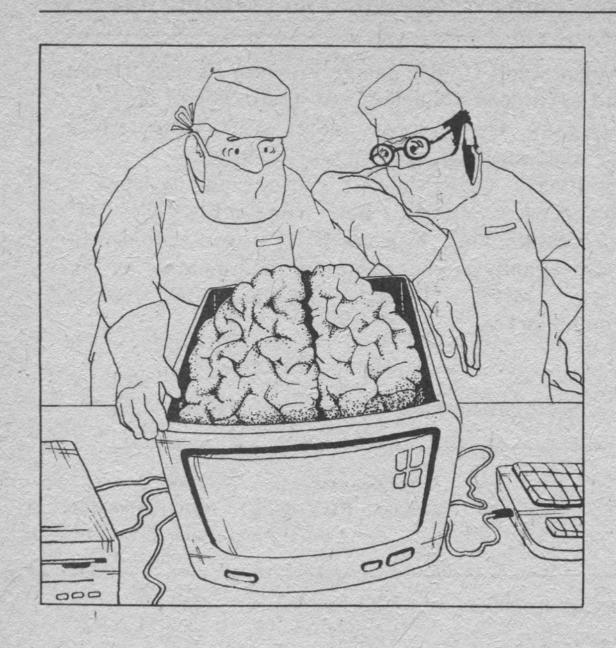
Для анализа электронных схем фирма Spectrum Software разработала два пакета MICROCAP (аналоговые схемы) и MICROLOGIC (цифровые схемы). Оба пакета включают подсистему вычерчивания электронных схем с применением стандартных обозначений и подсистему моделирования. С помощью подсистемы моделирования к схеме можно программным образом приложить различные питающие напряжения, входные сигналы постоянного и переменного напряжения, исследовать устойчивость схем, провести анализ переходных характеристик и динамических процессов. В библиотеке MICROCAP имеется широкий класс моделей активных и пассивных элементов электронных схем, в том числе различных видов диодов, биполярных и полевых транзисторов и др. Все характеристики исследуемой схемы можно наблюдать на экране дисплея, который в этом случае работает как многолучевой осциллограф. Пакет MICROLOGIC позволяет одновременно создавать и анализировать до 9 цифровых блоков, каждый из которых может содержать до 200 вентилей. Блоки можно анализировать как в автономном режиме, так и соединенными в схему. В пакете предусмотрена богатая библиотека стандартных логических элементов. Кроме того, возможно дополнительно определять до 36 логических элементов пользователя, каждый из которых может иметь до 36 каналов 256-битовых данных. При исследовании схем можно задавать до 10 различных форм цифровых сигналов. Попытка оценить оба пакета приводит к выводу, что, с одной стороны, их можно рассматривать как составную часть АСНИ, принимая во внимание научную область применения пакетов — анализ электронных схем, а с другой — как САПР, учитывая оптимальный синтез электронных схем. В дальнейшем разработанная схема может поступить в качестве исходного образца для САПР печатных плат.

Фирма DNASTAR разработала два пакета прикладных программ для исследователей, занимающихся биохимией белка. Первый пакет является аналитическим и служит для решения таких задач, как построение вторичной структуры белков, определение оптимального плана изучения ДНК для ряда специальных белков и протеинов, молекулярного веса белков, кривых титрации и других аналитических функций. Второй пакет Geneman помогает осуществлять поиск информации в базах данных по белкам и генетике, а также формировать свои собственные проблемно-ориентированные БД. Вместе с пакетом поставляются две базы данных — Gen Bank и Protein Identification Resource, на основании которых пользователи могут создавать свои собственные, ориентированные на конкретные приложения БД. Имеются также интерфейсы для работы с другими базами данных по биохимии белка.

Фирма Numelec поставляет рабочую станцию HISTOPERICOLOR, построенную по модульному принципу, для исследования клеточных структур. В комплект станции входят встроенная 16-битовая микроЭВМ с наращиваемой оперативной памятью до 16 Мбайт, внешней памятью на магнитных дисках от 10 Мбайт до 1 Гбайта, два графических дисплея, один из которых — цветной повышенного разрешения, цветная видеокамера, подключаемая к электронному микроскопу. Управление системой осуществляется с помощью интерактивного полиэкранного меню. Система позволяет производить анализ клеточных структур, создавать и накапливать цифровые видеоизображения в графической базе данных, автоматически вести регистрационный журнал экспериментов и т.п.

Как можно убедиться, автоматизированные системы научных исследований отличаются широким разнообразием. В создании таких систем наметились два направления. Одно направление связано с разработкой специализированных систем (рабочих станций), предназначенных для решения узкого круга задач. Второе направление, отличаясь большей массовостью, связано с разработкой универсальных пакетов широкого назначения. При этом между двумя направлениями существует определенная закономерность — по мере совершенствования ПК все больше специализированных функций переходит в разряд массовых.

Г.Чоговадзе



Нейрокомпьютеры японских фирм компенсируют недостатки существующей вычислительной техники. Они сокращают затраты на решение задач распознавания изображений, текста, речи, анализа банковских операций и ряда других. Подключив нейроплату к персональному компьютеру, вы получите нейрокомпьютер!

Последние модели персональных нейрокомпьютеров фирм NIHON DENKI и FUJITSU

Как известно, нейрокомпьютеры — суть попытка заставить компьютер обрабатывать информацию по принципу функционирования нервной системы человека, что в случае успеха сулит необычайные возможности вычислительной техники и в результате — господство на мировом рынке (см. КомпьютерПресс № 9'90).

Японские фирмы Nihon Denki и Fujitsu в 1988-89 годах поставили на рынок персональные нейрокомпьютеры Neuro-07 и FMR моделей 50, 60 и 70 (с подключаемой нейроплатой и программным обеспечением NEUROSIM/L) соответственно.

Nihon Denki считает, что направлениями эффективного практического применения нейрокомпьютеров являются следующие:

а) решение проблем, которые не могут быть формализованы и описаны математически;

б) реализация систем, характеризующихся свойствами адаптивности и расширяемости, а также способностью обучаться;

в) решение задач, имеющих значительную неопределенность (расплывчатость);

г) управление компьютерными сетями;

д) создание сверхнадежных компьютеров.

По мнению фирмы реально нейрокомпьютеры найдут практическое применение в обучаемых экспертных системах для прогнозирования и для управления процессами, в диагностике неисправностей машин и механизмов, при выделении сигналов в задачах гидроакустики, а также при анализе акустических сигналов живых организмов, сжатии видео- и аудиоинформации, синтезировании звуков и особенно при создании роботов с визуальной системой контроля и при управлении их движением (прикладные исследования нейронного моделирования).

В свою очередь фирма Fujitsu ориентирует свои персональные нейрокомпьютеры на работу в таких областях, как: диагностирование неисправностей машин и механизмов, обработка сигналов, распознавание текстовой информации и речи, а также в банковском деле при операциях с переводными векселями.

Информация об указанных нейрокомпьютерах вызвала живой интерес специалистов более чем ста фирм и организаций. Запросы поступили от банков, медицинских учреждений, предприятий по производству медицинского оборудования, машиностроителей, акционерных компаний, разработчиков программного обеспечения для автоматизированных производств.

Персональный нейрокомпьютер Neuro-07 Фирма Nihon Denki

По мнению Nihon Denki нейрокомпьютеры как особый класс устройств обработки информации способны компенсировать ряд недостатков традиционных (цифровых) вычислительных средств. Последним, как известно, свойственно осуществлять обработку информации в соответствии с установленной иерархией, которая представляется в виде совокупности предварительно разработанных алгоритмов. Алгоритмы требуют подробного описания задачи, определяющего последовательность обработки поступающей информации. Нейрокомпьютер позволяет отказаться от такой жесткой иерархии обработки информации. В результате, реализация, например, задачи распознавания изображений на нейрокомпьютере менее трудоемка, чем на традиционном современном компьютере.

В настоящее время в продаже имеется нейрокомпьютер Neuro-07, разработанный научно-исследовательским институтом "C&C" фирмы Nihon Denki Information Technology (NEC-IT, Миното, префектура Токио).

NEURO-07 представляет собой персональный компьютер серии PC-9800 с подключенной к нему специальной нейроплатой. Структура программных и технических средств нейрокомпьютера Neuro-07 приведена на рис.1.

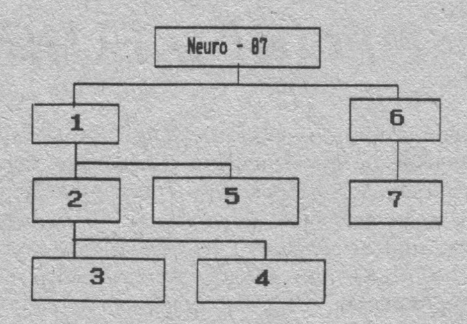


Рис. 1. Архитектура нейрокомпьютера Neuro-07:
1 — программные средства; 2 — решение: "способ"-"реальность"; 3 — пакет стандартных программ ("способ"); 4 — пакет нейропрограмм ("реальность"); 5 — библиотека языка Си на базе ImPP-процессора; 6 — технические средства; 7 — платы ImPP.

В продаже имеются платы нейросредств (neuro engine board) (ImPP: Image Pipeline Processor — конвейерный процессор изображений) стоимостью 180 тыс.йен; программные средства библиотеки языка Си на основе ImPP-плат — 30 тыс.йен; пакетов нейропрограмм (neuro soft package) и стандартных программ — общей стоимостью 470 тыс.йен. Таким образом, пользователи, имеющие компьютеры серии PC-9800, могут получить нейрокомпьютер, доплатив 700 тыс.йен.

Рассмотрим технические средства. Так называемый ImPP-процессор (ImPP: Image Pipeline Processor) является по сути дела конвейерным процессором обработки изображений. Он присоединяется к стандартному варианту плат компьютеров серии PC-9800.

ImPP-плата (ImPP-board) состоит из четырех параллельно соединенных БИС типа mPD 7281 (mPD: microprocessor data). Каждая БИС имеет закольцованную конвейерную конструкцию, состоящую из пяти структур MIPS (Metal Insulator Piezoelectric Semiconductor).

Таблица 1

Основные характеристики Neuro-07

	Максимальные (модель PC + ImPP)	Для распознавания знаков (модель PC)
Количество нейронов	82000	220
Число связей	246000	7000
Скорость	216000	196000
Время обучения	3,5 ч	160 ч
Скорость распознавания	60 зн/с	4 зн/с

Указанные БИС называются конвейерными процессорами изображений и разработаны для высокоскоростной обработки видеоинформации, причем по своему быстродействию они сравнимы с нейронными сетями.

Математическое обеспечение нейрокомпьютера представлено пакетом стандартных программ ("Способ"), функционирующих в среде операционной системы MS-DOS, и пакетом нейропрограмм ("реальность"), зашитым в платы конвейерного процессора изображений, а также прикладным программным обеспечением на базе библиотеки языка Си.

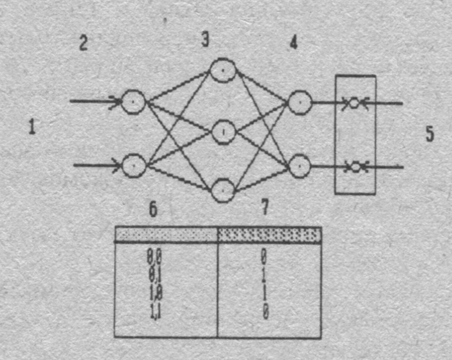


Рис. 2. Схема нейросети логических операций XOR (исключающее ИЛИ) Neuro-07:
1 — вводимый образец ; 2 — уровень ввода; 3 — промежуточный уровень; 4 — уровень вывода; 5 — сигнал от обучающего; 6 — ввод; 7 — вывод.

В Neuro-07 реализована модель нейронной сети, называемая "обратной передачей полномочий" (правило обратного распространения обучения). Логические операции нейросети компьютера Neuro-07 разбиты на следующие уровни: ввода, промежуточного уровня и вывода, что показано на рис.2. Все уровни состоят из большого количества нейронов, каждый из которых связан со всеми нейронами соседних уровней. Обязательным условием функционирования нейрокомпьютера является его обучение. Оно происходит следующим образом: после ввода эталонных данных на промежуточном и выходном уровнях последовательно производят вычисления. Затем проводят аналогичные действия в обратном направлении с целью минимизации ошибок и изменяют значимость каждой нейронной связи. Обучение (обычно несколько циклов) идет до тех пор, пока не будет достигнута заданная величина ошибки.

Рис.3 иллюстрирует обучение нейрокомпьютера распознаванию эталонного цифрового изображения цифры "2". В комплект Neuro-07 входят пять видов наборов изображений чисел от 0 до 9, что дает возможность изменять коэффициент распознавания и тенденцию обучения нейрокомпьютера при изменении коэффициентов направленности S-образной кривой в изображении цифры, пороговых значений коэффициента забывания, поправочного коэффициента, коэффициента забывания нагрузки, поправочного коэффициента нагрузки, структуры сети нейронов и числа их промежуточных слоев.

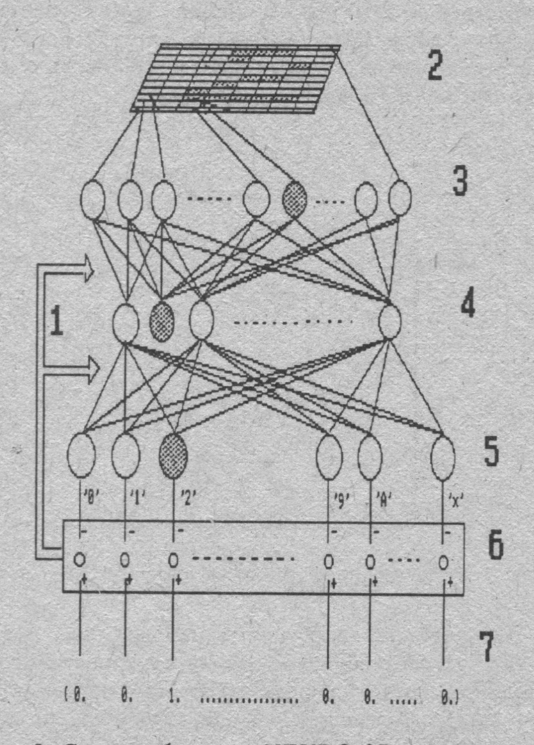


Рис. 3. Схема обучения NEURO-07 распознаванию простых цифровых изображений:
1 — задание погрешности; 2 — вводимое изображение; 3 — уровень ввода; 4 — промежуточный уровень; 5 — уровень вывода; 6 — вычисление погрешности; 7 — ввод от учителя.

Neuro-07 распознает каллиграфические знаки японского алфавита каны при их считывании телекамерой. Обучение происходит путем ввода изображения знаков с экрана монитора с помощью "мыши". Нейрокомпьютер способен также распознавать почти все знаки японского алфавита каны, которые имеют искажения в своем написании.

Нейрокомпьютер обладает несколько лучшей степенью распознавания печатных знаков по сравнению с традиционными современными компьютерами. Некоторые сравнительные характеристики приведены ниже в табл. 2.

Таблица 2

Вид распознаваемой информации		Точность распознавания (погрешность): нейрокомпьютер	Точность распознавания (погрешность): обычный компьютер
Печатные знаки	Листинг 12 шрифтов с 76 буквенно-цифр.знаками	99,95% (0,05%)	99,67% (0,33%)
	Листинг 62 буквенно-цифровых знака точечной печати	99,8% (0,2%)	98,5% (1,5%)
Речь	Произвольный рассказчик (10 цифр)	99,3% (0,7%)	98,9% (1,1%)

Возможности Neuro-07 ограничены распознаванием отдельно арабских цифр и знаков японского алфавита каны, а распознавание произвольного текста, содержащего иероглифы, пока неосуществимо.

Разработкой программного обеспечения Neuro-07 в течении года занимались 5 человек из Nihon Denki и двое из NEC-IT. К середине 1989 года было продано около 180 экземпляров Neuro-07, причем 60% купили научно-исследовательские организации и учреждения; 30% — исследовательские и проектные отделы промышленных предприятий; 10% — обычные промышленные предприятия и частные лица.

Касаясь перспектив нейрокомпьютеров, фирма Nihon Denki считает, что они не будут вытеснять обычные компьютеры, скорее всего те и другие будут мирно сосуществовать и эффективно дополнять друг друга. Neuro-07 рассматривается как базовая модель для перспективных моделей — Neuro-09 и Neuro-11.

Персональные компьютеры фирмы Fujitsu

Фирма Fujitsu заявила о достижении десятикратного превосходства технических характеристик своих персональных нейрокомпьютеров над аналогичными компьютерами фирмы Nihon Denki. Нейрофикацию компьютеров фирма осуществляет описанным выше способом, а именно, подсоединяя к существующим ПК

специальные нейроплаты и соответствующее программное обеспечение (ПО). Однако превосходство обеспечивается за счет своего ноу-хау.

В состав нейрокомпьютера входят выпускаемый фирмой Fujitsu ПК серии FMR (моделей 50, 60 и 70), нейроплата и программное обеспечение NEUROSIM/L, обеспечивающее обучение нейросети. Схема обучения представлена на рис.4.

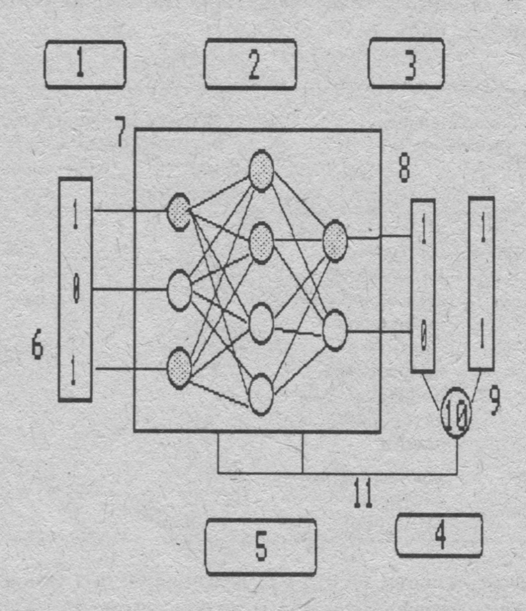


Рис. 4. Схема обучения нейрокомпьютера фирмы Fujitsu:
1 — предъявление исходной информации; 2 — вычисления; 3 — выдача решения; 4 — вычисление погрешности; 5 — корректировка "веса" каждого нейрона; 6 — вводимая информация; 7 — нейросеть; 8 — выходной результат; 9 — правильное решение; 10 — сопоставление; 11 — корректировка с учетом погрешности E.

Нейроплата ПК представляет собой выпускаемый фирмой Fujitsu сверхбыстродействующий (4 mips) процессор цифровой обработки сигналов модели MB 86232. Нейроплата имеет собственную память емкостью 4 Мбайта, что позволяет моделировать нейросеть, состоящую из 1000 нейронов. Модель нейросети — иерархического типа. Она включает в себя входной, промежуточный и выходной уровни. Благодаря наличию собственной памяти обеспечивается возможность создания четырехуровневой сетевой структуры нейросети, состоящей из входного, выходного и двух промежуточных уровней. Путем изменения количества нейронов на каждом уровне можно осуществлять разнообразные виды моделирования.

Fujitsu ориентируется на трехуровневые и четырехуровневые модели нейросети. Модели, содержащие пять и более уровней, теоретически могут быть сведены к четырехуровневой модели.

Для повышения качества своего нейрокомпьютера фирма разработала оригинальные алгоритмы его обучения и соответствующее программное обеспечение, которые названы следующим образом: 1) метод виртуального импенданса; 2) метод скорректированного обучения; 3) метод расширения обучения.

Сущность метода виртуального импенданса (см. рис.5) заключается в том, что после начала обучения оно циклически повторяется, при этом проводится корректировка "веса" W нейронов с целью достижения заданной погрешности E. Формула корректировки веса выглядит следующим образом:

$dW = -e*dE/dW$,

где e — константа.

Это, однако, совсем не означает, что уменьшение погрешности будет происходить постоянно в соответствии с какой-то заданной кривой. Так, в частности, левая часть изображенной на рис.5 кривой отражает непрерывное уменьшение погрешности, однако, в результате изменения весовой пропорции, в некоторой точке происходит перегиб кривой в сторону увеличения погрешности (локальный максимум в левой части кривой). Поэтому, основываясь на каком-либо типовом прогнозе характера изменения погрешности, обеспечивают подобное уменьшение погрешности по убывающей кривой, быстро стремящейся к точке E. Благодаря этому, как утверждается, скорость обучения возрастает в 5-10 раз. В общих чертах специалисты фирмы сформулировали метод виртуального импенданса как "метод, построенный на аналогии с механическими колебательными системами".

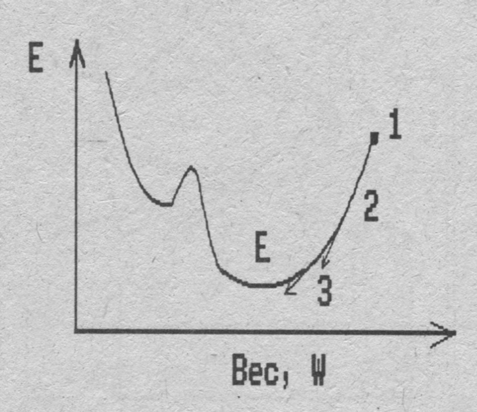


Рис. 5. Качество обучения нейрокомпьютера (алгоритм быстрого обучения по методу виртуального импенданса; методу скорректированного обучения) :
1 — начало обучения; 2 — малые корректировки веса; 3 — окончание обучения.

Метод скорректированного обучения заключается в уменьшении непродуктивного периода обучения. Это достигается тем, что из совокупности используемых в процессе обучения примеров используются те, которые

относятся к завершающей стадии обучения, что обеспечивает ускорение обучения примерно в 1,2 раза.

Что касается метода расширения обучения, то он состоит в том, что совокупность учебных примеров предварительно разбивается на несколько групп и дальнейшее обучение проводится с использованием этих групп. Благодаря этому, в тех случаях, когда в процессе обучения дополнительно вводятся новые учебные примеры, то на новом (повторном) цикле обучения отпадает необходимость в изучении всех примеров с самого начала. В итоге за счет применения методов скорректированного обучения и расширения обучения достигается четырехкратное повышение скорости обучения нейрокомпьютера.

Фирма Fujitsu продемонстрировала возможности своего нейрокомпьютера при работе в системе диагностирования неисправностей машин и механизмов, которое осуществляется на основе анализа их вибрационных сигналов, а также в банковском деле при оценке деятельности 12 фирм, производящих электротехническое оборудование, используя классификацию их переводных векселей. В последней задаче моделировалось финансовое положение этих фирм на основе обучения нейрокомпьютера по финансовым показателям аналогичных электротехнических фирм. В качестве обучающей использована информация: о ценных бумагах, находящихся в обращении и в пассиве; о доле собственного капитала; нормах прибыли от объема продажи, на совокупный и собственный капитал, прироста дохода; о коэффициентах оборачиваемости капитала и др. Информация сформирована стандартным способом, а именно, с помощью пакетов LOTUS 1-2-3 и dBASE III. Результаты моделирования с помощью нейрокомпьютера показали их хорошее совпадение с фактическими данными, опубликованными в прессе. Отметим, что результаты моделирования могут быть представлены также с помощью LOTUS 1-2-3.

Цены на нейрокомпьютеры

Цена нейрокомплекта ПК+ПО фирмы Nihon Denki составляет 180+470 = 650 тыс.йен, а фирмы Fujitsu — 730+250 = 980 тыс.йен, однако коэффициент затраты/технические возможности у последней ниже.

Некоторые советы по практическому использованию нейрокомпьютеров

Совет 1. Используйте нейрокомпьютеры в качестве инструмента для объединения моделей явлений и процессов в единую систему с целью получения более достоверной информации об изучаемом предмете.

Совет 2. Используйте нейрокомпьютеры в сочетании с экспертными системами и коммерческими пакетами (оболочками) для их построения, например, ESHEL L/FM. Имейте в виду, что по сравнению с экспертными системами нынешние нейрокомпьютеры менее точны, однако это зависит от качества их обучения.

А.Стебунов

По материалам:

Computopia, 1989, No.6, p.p. 56-62,
Computopia, 1989, No.7, p.p. 120-124,
Computer Today, 1990, No.35, p.p.26-36.

Компания Dragon Systems из Массачусетса выпустила версию программы распознавания речи Dragon Dictate, работающую на машинах с архитектурой Micro Channel. Dragon Dictate-MCA будет иметь те же режимы, что и выпускаемый ныне вариант для IBM AT, но плата распознавания речи выполнена в стандарте MCA.

Комбинация платы и программы, продаваемая за 9000 долларов, может распознавать одновременно до 30 тысяч слов при словаре в 80 тысяч. По заявлению руководителя компании, такая система будет применяться главным образом в тех случаях, когда с компьютером работает инвалид, или же когда пользователь не желает, либо в силу занятости не может использовать клавиатуру для ввода информации.

Dragon Dictate может преобразовывать речь в текст и вводить его в такие программы, как WordPerfect или Lotus 1-2-3 при условии, что говорящий будет делать паузы между словами не менее четверти секунды. По заявлению фирмы, скорость ввода может достигать 30-40 слов в минуту.

Newsbytes News Network, December 24, 1990

Калифорнийская фирма Stac Electronics выпустила программу Stacker, которая, по утверждению фирмы, позволяет сжать информацию на жестком диске ровно в два раза, т.е. на диске объемом 20 Мбайт можно записать 40 Мбайт информации. Существует две версии этой программы: для "обычного" сжатия информации — чисто "программный" продукт стоимостью 129 долларов, и для "промышленного" ускоренного сжатия — программа с платой сопроцессора, устанавливаемой в гнездо AT половинного размера, — 229 долларов. Для работы этой программы требуется операционная система, начиная с MS-DOS 3.0.

PC Magazine, December, 25, 1990

Фирма Sony объявила о выпуске 18-фунтовой (8 кг) RISC рабочей станции с жестким диском емкостью 406 Мбайт и скоростью работы 17 миллионов операций в секунду. Цена изделия — 14 тысяч долларов.

Federal Computer Week, December 17 1990

Лазерный принтер, бойко и легко выплевывающий отпечатанные почти с типографским качеством листы в выходной лоток, является устройством, сложностью и слаженностью действия напоминающим современный завод в миниатюре. Увидев это чудо XX века, Иван Федоров был бы потрясен.

ЛАЗЕРНЫЙ ПРИНТЕР

Объединенные в лазерном принтере достижения в областях прецизионной механики, оптики, лазерной технологии, методов обработки изображений и микропроцессорного управления по праву ставит его в первые ряды технического прогресса.

Чтобы получить отпечаток, лазерный принтер должен принять из компьютера сигнал, преобразовать его во множество команд, управляющих движением лазерного луча к перемещению бумаги, а также работой множества дополнительных (но так необходимых!) узлов.

Так как же работает лазерный принтер? Прежде всего несколько слов о принципе действия. В лазерных принтерах используется электрографический принцип создания изображения (такой же, как в копировальных машинах фирмы Xerox).

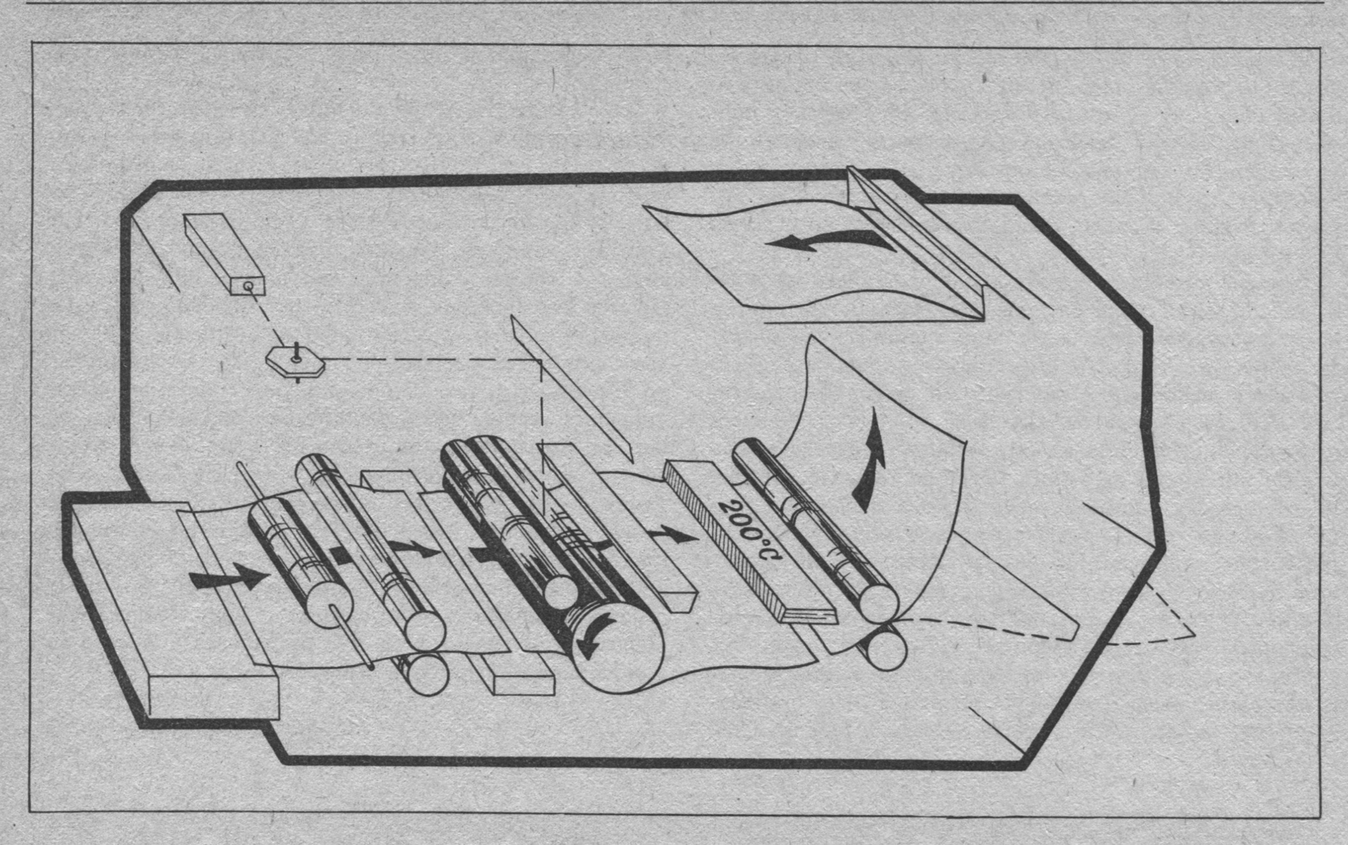
Сердцем лазерного принтера является фотопроводящий цилиндр (organic photoconducting cartridge), который часто называют печатающим барабаном. С помощью барабана производится перенос изображения на бумагу. Он представляет собой металлический цилиндр, покрытый тонкой пленкой фотопроводящего полупроводника, обычно оксидом цинка или чем-либо подобным. Поверхности этого покрытия можно придать положительный или отрицательный заряд, который сохраняется на поверхности, но только до тех пор, пока барабан не освещен. Если какую-либо часть барабана проэкспонировать, то покрытие приобретет проводимость и заряд стечет с освещенного участка, образовав незаряженную зону. Данный момент очень важен для понимания принципа работы лазерного принтера.

Следующей важной его частью является лазер и прецизионная оптико-механическая система, перемещающая луч.

Малогабаритный лазер генерирует тонкий световой луч, отражающийся от вращающегося зеркала (как правило, шестигранного), разряжает положительно заряженную поверхность барабана. Чтобы получить изображение, лазер включается и выключается управляющим микроконтроллером. Вращающееся зеркало разворачивает луч в строку на поверхности печатающего барабана. Все это вместе создает на его поверхности строку скрытого изображения, в котором те участки, которые должны быть черными, имеют один заряд, а белые — противоположный. После формирования строки изображения специальный прецизионный шаговый двигатель поворачивает барабан так, чтобы можно было формировать следующую строку. Это смещение равняется разрешающей способности принтера и обычно составляет 1/300 дюйма (по поверхности), то есть 0.1 мм. Этот этап печати напоминает построение изображения на экране телевизора или монитора.

Но каким образом на поверхности барабана появляется заряд, необходимый для создания изображения? Для этого служит тонкая проволока или сетка, называемая “коронирующим проводом“. Но почему “коронирующий“? Дело в том, что на этот провод подается высокое напряжение, вызывающее возникновение светящейся ионизированной области вокруг него, который и называется короной и придает барабану необходимый статический заряд.

Итак, на барабане сформировано изображение в виде статического заряда и незаряженных участков. Что дальше? Дальше барабан проходит мимо валика, подающего из специального контейнера черный красящий порошок — тонер. Частички тонера, заряженные положительно, прилипают только к нейтральным участкам, отталкиваясь от положительно заряженных. Это похоже на то, как на экране телевизора собирается пыль.



Небольшое замечание: здесь речь идет о принтерах типа Hewlett-Packard LaserJet. Однако существует и другой метод формирования изображения. Он используется в принтерах Epson и других подобных, использующих двигатель фирмы Ricoh. В этих принтерах разряжаются участки, которые должны быть белыми. В этом случае тонер, заряженный отрицательно, притягивается к положительно заряженным участкам барабана. Отпечатки, изготовленные на таких принтерах, имеют едва уловимые различия в качестве: при использовании первого способа достигается лучшая передача деталей, а при работе со вторым — более качественные черные области.

Следующим этапом является перенос тонера (а, значит, и изображения) на бумагу. Бумага вытягивается из подающего лотка и с помощью системы валиков перемещается к печатающему барабану. Перед самым барабаном бумаге сообщается статический заряд с помощью еще одного коронирующего провода, подобного тому, что используется для подготовки барабана к экспонированию. Затем бумага прижимается к поверхности барабана. Заряды разной полярности, накопленные на поверхности бумаги и на поверхности барабана, вызывают перенос частиц тонера на бумагу и их надежное прилипание к последней. После переноса тонера бумага покидает поверхность барабана.

При этом валики продолжают перемещать бумагу к выходному лотку принтера. Следующим звеном принтера, встречающим бумагу с изображением на этом пути, является узел фиксации изображения. Тонер содержит вещество, способное легко плавиться. Обычно это какой-либо полимер или смола. При нагревании до 200-220 градусов и повышении давления порошок расплавляется и намертво соединяется с поверхностью бумаги. Только что вышедшие из принтера листы — теплые, а слишком нетерпеливый пользователь, хватающий появившийся отпечатанный листок, рискует обжечь пальцы.

Далее бумага протаскивается к выходному лотку. При этом, если листы выводятся напрямую, верхним в стопе отпечатков оказывается последний лист. Многие принтеры, однако, переворачивсают бумагу лицом вниз, складывая стопу в правильном порядке — то есть верхним будет первый лист, нижним — последний.

Отпечаток готов, осталась не рассмотренной последняя важная позиция — очистка барабана. При переносе изображения на бумагу не все частички тонера прилипают к ней и небольшое количество их остается на барабане. Их наличие приведет к появлению на следующей апечатаемойстранице грязи. Поэтому остатки тонера удаляются с поверхности барабана специальным чистящим узлом. Теперь цикл закончен, барабан заряжается вновь — и можно печатать следующую страницу.

Важным является устройство управления, как правило, микроконтроллер на базе процессоров 68000 или 68020 фирмы Motorola. Контроллер обслуживает порты, оперативную память, осуществляет диагностику принтера, выдает сообщения на панель управления, эмулирует различные стандарты подключения и, конечно, выдает десятки сигналов, управляющих всеми узлами принтера.

Первые лазерные принтеры, появившиеся в 1984-85 годах, были столь сложны, что разработки приемлемого программного обеспечения пришлось дожидаться почти два года. До этого времени единственным способом обеспечения доступа ко всему множеству технических возможностей новых принтеров являлось использование специальных команд — последовательностей символов, один вид которых вызывал страх у неискушенных пользователей. Первые программы, решив в какой-то мере проблемы распечатки текстов, не позволяли пользователю вычерчивать прямые линии или прямоугольники, наносить тени или показывать оттенки, а также использовать для распечатки текстов различные гарнитуры шрифты. Поэтому появилось несколько основных стандартов обмена с принтерами и программные драйверы для работы в этих стандартах. Два наиболее значимых — язык PCL фирмы Hewlett-Packard и язык PostScript, разработка фирмы Adobe.

Эти стандарты скорее дополняют друг друга, чем конкурируют между собой. Первый отличается тем, что работает с побитными шрифтами и растрированной (еще в компьютере) графикой. Это позволяет работать только со шрифтами ограниченного размера (так как шрифты больших размеров требуют значительных объемов оперативной памяти в принтере). Другой сложностью является то, что каждый кегль шрифта должен разрабатываться отдельно. Второй язык позволяет работать со шрифтами кеглем от 0.5 до 999 пунктов, так как используются математические описания формы букв, конкретное расположение точек на отпечатке рассчитывается в принтере. Кроме того, графическое изображение также описывается математически, а принтер оптимальным образом строит результирующее изображение. PostScript оставляет простор для роста качества — он позволяет работать с любым разрешением — выводное устройство всегда стремится полностью использовать свои возможности. Недостатком является то, что разработка шрифтов является значительно более трудоемким делом.

И.Вязаничев, Б.Молчанов

Уважаемые читатели, в октябрьском выпуске нашего журнала мы открыли постоянную ежеквартальную рубрику "Журнальный киоск" московской фирмы "Бюро коммерческой связи". Повторно публикуя этот материал, мы выражаем уверенность в том, что подобная услуга заинтересует серьезных специалистов в области создания, использования и реализации компьютерной техники и программного обеспечения. Особого внимания заслуживает уникальность и доступность данной услуги, т.к. впервые валютные издания по вычислительной технике стало возможным приобрести за отечественные рубли. Редакция КомпьютерПресс приносит свои извинения за неверно указанный в предыдущей публикации почтовый индекс "Бюро коммерческой связи".

Журнальный киоск

фирмы "Бюро коммерческой связи"

Уважаемые товарищи! Впервые приступая к распространению в СССР лучших иностранных изданий по вычислительной технике, мы отдаем себе отчет в двойственности ситуации.

С одной стороны, нам приятно помочь талантливым отечественным программистам, инженерам, специалистам по вычислительной технике попасть в огромный мир западного компьютерного бизнеса и науки. Предлагаемые журналы содержат большое количество фирменных рекламных буклетов и коммерческих предложений, а количество изданий производит неизгладимое впечатление на неизбалованного отечественного потребителя.

С другой стороны, возможности нашей фирмы в настоящее время вряд ли смогут удовлетворить потенциальный спрос. Правда, расширение поставок во многом будет зависеть от вашей заинтересованности.

Для оценки спроса на издания и их потребительской стоимости мы предлагаем первый раз осуществить их реализацию в виде заочного аукциона. Ниже публикуется каталог изданий, которыми в настоящее время располагает фирма. Возможно, в скором времени он будет расширен.

СЕГОДНЯ В ПРОДАЖЕ

№	Наименование	1990 год	Цена одного журнала
1.	**"Amiga world"**	**июнь, июль, август**	**$3.95**
2.	**"PC world"**	**май, июнь, июль, август**	**$2.95**
3.	**"PC magazine"**	**май, июнь, июль, август**	**$2.95**
4.	**"Compute"**	**июнь**	**$2.95**
5.	**"Mac world"**	**июнь, июль, август**	**$2.95**
6.	**"Unix review"**	**июнь, июль**	**$3.95**

Просим Вас направить по адресу 115408, Москва, а/я 1 заказ, в котором сообщить:

а) почтовый индекс, адрес, название организации и ФИО заказчика, желательно телефон;

б) наименование, месяц издания, количество приобретаемых журналов, а также Вашу оценку их потребительской стоимости в рублях;

в) Ваше желание оформить постоянный заказ на те или иные издания.

О принятии к исполнению заявок Вам будет направлено специальное извещение. Рассылка изданий производится в пакетах после предоплаты заказа в соответствии с направленным заказчикам извещением.

В дальнейшем порядок оформления заказов значительно упростится. Мы планируем сократить срок доставки изданий подписчикам до I квартала с момента их выхода в свет.

Внимание! *В связи с ограниченым фондом журналов в первую очередь удовлетворяются запросы заказчиков, оперативно выразивших свою заинтересованность в регулярной поставке журналов.*

Наши программисты не приучены читать фирменную документацию в силу известных причин, из которых главные — добывание программных продуктов пиратскими путями и незнание английского языка. Поэтому вокруг многих простых вопросов возникает ореол таинственности. Один из них — как красиво представляемая на экране информация хранится на диске? Понимая трудности приобретения пакетов, мы предлагаем вашему вниманию описание структуры файла базы данных СУБД dBASE. Этот файл имеет расширение имени .DBF.

Структура файла DBF

Файлы DBF имеют стандартную структуру во всех типах программного обеспечения микрокомпьютеров. Подавляющее большинство электронных таблиц, СУБД и интегрированных пакетов имеют средства экспорта и импорта этих файлов. На то есть две причины, первая из которых — популярность пакета dBASE, а вторая — простота организации этих файлов. Заголовок файла имеет фиксированную длину, а список дескрипторов, следующий непосредственно за ним, — переменное число дескрипторов (см рис.1). Далее в файле содержатся записи фиксированной длины, информация в которых представлена в коде ASCII. Каждая запись содержит заголовочный байт, — символ пробела (десятичный код ASCII — 32), за исключением заголовочных байтов удаленных записей. Удаленные записи помечаются звездочкой.

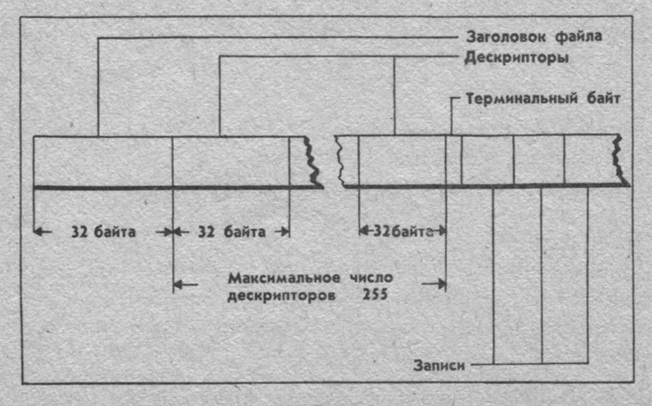


Рис.1

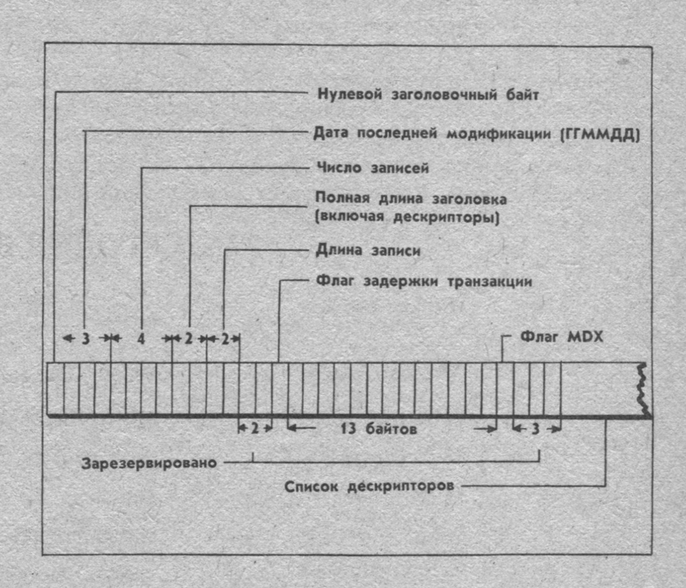


Рис.2

Заголовок файла DBF всех версий, начиная с dBASE III, имеет постоянную длину 32 байта (рис.2).

Нулевой байт заголовка имеет структуру битовой карты (рис.3) и содержит номер версии и информацию о состоянии (подключен ли текстовый файл DBT-memo).

Следующие три байта содержат шестнадцатиричную дату последнего обновления в формате ГГММДД. После даты расположен счетчик записей (включая удаленные) в виде длинного целого числа без знака (четыре байта). Следующие два байта описывают в

виде обычного целого числа без знака совокупную длину заголовка и списка дескрипторов полей. Следующее число без знака определяет длину записей файла. Далее идут два зарезервированных байта и байт, имеющий значение 0-1, определяющий задержку по транзакции (только для dBASE IV). Байты с 15 по 27 зарезервированы. Байт 28 используется только в dBASE IV и указывает, подключен ли к файлу множественный индекс — файл MDX. И, наконец, байты с 29 по 31 также зарезервированы.

Сразу за заголовком следуют дескрипторы полей. Как и заголовок, в версиях выше dBASE II каждый дескриптор имеет постоянную длину 32 байта (рис.4). Максимальное число дескрипторов на один файл — 255. (В Clipper это число достигает 512).

Первые одиннадцать байтов каждого дескриптора содержат имя поля, неиспользованные символы заполняются нулями. В двенадцатом байте указан тип поля (C,L,N,M или F). Следующие четыре байта также заполнены нулями и зарезервированы. Шестнадцатый байт описывает длину поля (включая десятичные разряды и десятичную точку), тогда как семнадцатый байт содержит только количество десятичных разрядов. Следующие тринадцать байтов зарезервированы. И, наконец, последний байт дескриптора является флагом тэга многоиндексного файла MDX в dBASE IV и имеет значение 0, если тэга нет, и 1 — если он есть.

Поле дескрипторов отделяется от записей файла терминальным символом — символом перевода строки (код ASCII — 10).

М.Михайлов

По материалам:
R.Freeland "The DBF divulged", Data Based Advisor, June, 1990.

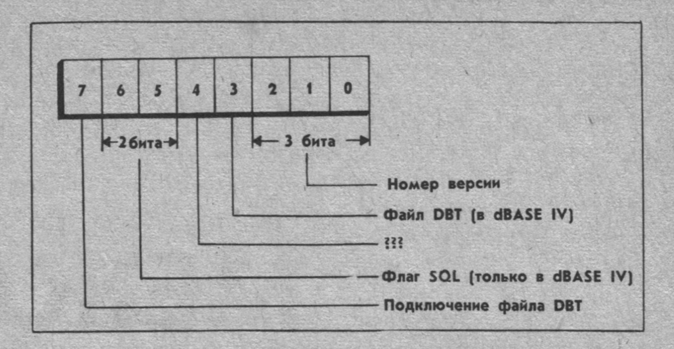


Рис.3

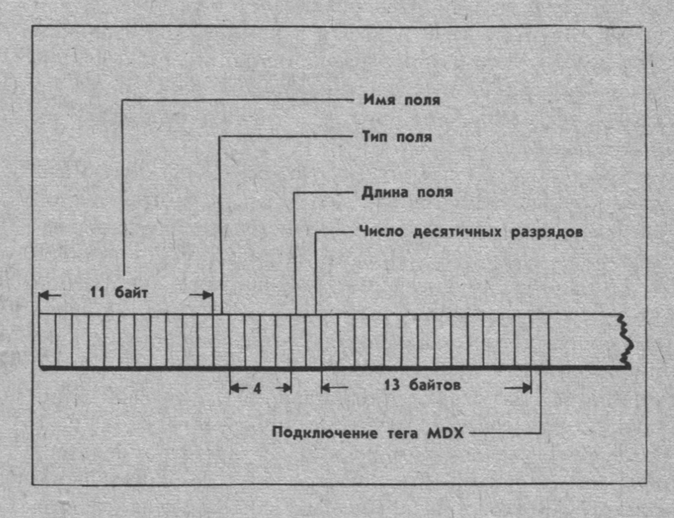


Рис.4

Читать со скоростью 2500 слов в минуту

Если вам необходимо сканировать тысячи страниц текста, то новая система оптического распознавания символов (ОРС) Parallel Reader фирмы Caere Corporation — это то, что нужно.

Система Parallel Reader позволяет считывать от 220 до 700 символов в секунду (2500 слов в минуту) с очень высокой точностью. Программа может распознавать шрифты размером от 6 до 72 пунктов на 11 западноевропейских языках и автоматически преобразовывать текст, набранный в несколько колонок. Она также распознает включенные в текст графические фрагменты и запоминает их как TIFF файлы.

Для нормальной работы системы Parallel Reader рекомендуется комплект, в который входят компьютер 80386SX с 40-Мбайтным жестким диском, VGA адаптер, четыре платы параллельных процессоров распознавания и собственно программа Parallel Reader, работающая под управлением Windows версии 3.0. Цена такого комплекта составляет 10995 долларов. Система поддерживает сканеры фирм Hewlett-Packard, Microtec и Fujitsu.

Desktop Magazine, October, 1990

26 февраля 1991 г. поступит в продажу новый портативный компьютер фирмы IBM. Выполненный на микропроцессоре 386SX, он предположительно будет весить 7.5 фунтов (3.4 кг). Как сообщается, новая машина с 60-мегабайтным жестким диском будет стоить около 5000 долларов.

Newsbytes News Network, December 26, 1990

МЕЖДУ ПРОЧИМ

Замена батарейки

В один прекрасный день вы включите компьютер и обнаружите, что система показывает неверные дату и время. Либо вы заметите, что у компьютера возникли проблемы с запоминанием конфигурации. Если какой-либо из этих симптомов проявляется регулярно, это скорее всего означает, что пора менять батарейку на плате часов/календаря.

Кое-кто может посчитать чрезмерной роскошью выдачу компьютером точных даты и времени при каждой загрузке системы. Однако наличие батарейки дает возможность машинам класса IBM PC/AT помнить существенную информацию, касающуюся аппаратной части системы. Данная информация сохраняется после выключения системы только благодаря подпитке специального ОЗУ от батарейки.

Многие платы часов поставляются с литиевой батарейкой размером с пятикопеечную монету (в ряде последних моделей компьютеров батарейка и встроенная схема часов монтируются на объединительной плате). Литиевые батарейки ни в коем случае нельзя перезаряжать. На плате часов есть специальный диод, который предохраняет батарейку от прохождения паразитного тока, способного ее перезарядить. Если такое все же случится, с компьютером могут произойти неприятности. Например, когда на одном компьютере вышел из строя упомянутый диод, это обошлось его владельцу в 800 долларов. Через батарейку прошел ток, в результате чего она взорвалась, прошив осколками встроенный модем, контроллер жесткого диска и объединительную плату.

Такие несчастья случаются редко, но все же полезно подстраховаться и заменить литиевую батарейку более надежной батарейкой стандарта R6 (элемент 316 или AA). Эта операция не требует больших усилий и средств. Если вы не хотите вносить какие-либо изменения в ваш компьютер, ниже вы найдете инструкцию по замене литиевой батарейки.

Если ваш компьютер имеет процессор 80286 или 80386, то, прежде чем приступить к замене батарейки, вам необходимо выполнить некоторую предварительную работу (если у вас компьютер класса PC или XT, просто следуйте инструкции). При удалении старой батарейки пропадет информация о конфигурации вашей системы. Поэтому, если вы не хотите вызывать специалистов для настройки компьютера, поработайте сначала несколько минут с программой установки, чтобы выяснить, какая содержащаяся в ней информация понадобится после того, как будет установлена новая батарейка.

В большинстве машин данную информацию можно получить двумя способами. Во-первых, можно запустить программу, которая обычно поставляется с компьютером и чаще всего имеет имя SETUP. В современных моделях нужно нажать определенную комбинацию клавиш при включении компьютера. Например, для машин, использующих Award BIOS, нужно нажать комбинацию клавиш Ctrl-Alt-Esc (при этом вызывается встроенная в BIOS компьютера программа установки). Итак, одним из доступных способов вы запустили программу установки. Когда на экране появятся текущие данные о конфигурации, сделайте твердую копию экрана либо просто перепишите эту информацию. Сохраните ее на будущее: когда-нибудь она может вам пригодиться, если у вас возникнут неожиданные проблемы с аппаратной частью. Теперь можно приступать к замене батарейки.

1. Позаботьтесь о защите системы. Положите на рабочую поверхность стола мягкую ткань. Прежде чем дотрагиваться до внутренних элементов стойки, прикоснитесь к корпусу компьютера, чтобы снять статическое электричество. Снимите кожух системного блока и найдите литиевую батарейку либо на плате часов/календаря, либо на объединительной плате. Если батарейка расположена на плате расширения, демонтируйте плату. Если батарейка расположена на объединительной плате, удалите все платы, затрудняющие доступ к ней.

2. При помощи отвертки удалите батарейку из держателя. Обычно в держателе располагается только одна батарейка. Если батареек оказалось две, значит вам понадобится в два раза больше элементов стандарта R6 и, дополнительно к ним, держатель для четырех батареек.

Если вы решили не менять тип элемента, тогда просто вставьте новую литиевую батарейку и установите плату часов на место. Не обращая внимания на оставшиеся пункты инструкции, запустите программу установки и настройте конфигурацию машины, после чего можете продолжить работу на компьютере.

3. Найдите места, в которых держатель припаян к плате. Посмотрите, к какому контакту припаяна ножка держателя, чакрывавшая батарейку. Возьмите фломастер и пометьте этот контакт знаком "+". При помощи паяльника удалите держатель литиевой батарейки. Постарайтесь не прикладывать паяльник к контактам надолго. Если вы перегреете плату, вы можете разрушить дорожки и плата станет негодной. Если вам не по душе работа с паяльником, обратитесь к кому-нибудь за помощью.

Альтернативное решение: оставьте держатель литиевой батарейки на месте, присоедините к нему держатель батарейки стандарта R6 при помощи проводов с "крокодильчиками", соблюдая полярность (соединяйте вместе одноименные полюса). Конечно, такое соединение менее надежно, однако это все же лучше, чем риск взрыва батарейки.

4. Не устанавливайте батарейки стандарта R6 в держатель, пока не закрепите его на плате. Припаяйте положительный провод держателя (обычно красного цвета) к контакту на плате, который вы пометили знаком "+". Припаяйте отрицательный провод держателя (обычно черного цвета) к оставшемуся контакту. Не берите припоя больше, чем это необходимо для пайки.

5. Установите батарейки в держатель, прочно зафиксировав их прозрачной липкой лентой. Установите плату в компьютер и осторожно положите батарейки в безопасное место.

Не бросайте использованные литиевые батарейки в огонь. Это может привести к взрыву и выделению токсичных газов.

Включите компьютер и установите дату и время. Для машин на базе процессоров 80286 и 80386 вам придется заново ввести информацию о конфигурации машины с помощью программы установки, воспользовавшись предварительно сделанной вами копией этой информации. Новые батарейки смогут надежно работать как минимум в течение года, после чего их следует заменить новыми.

Увеличение размера окружения

У многих пользователей бывает момент, когда после очередного усовершенствования операционной среды на экране появляется сообщение "Out of environment space". Причем, это может произойти и при выполнении команды, введенной с клавиатуры, и при работе командного файла. При этом файл AUTOEXEC.BAT имеет небольшой объем, а размер свободной оперативной памяти составляет порядка 600 Кбайт! В чем же проблема?

Дело в том, что блок, отведенный под запись параметров окружения, имеет определенный размер, по умолчанию равный 160 байт для MS-DOS 3.30 и 128 байт для более ранних версий этой операционной системы. Конечно, в такой объем много не запишешь. Заметим, что параметры среды можно вывести на экран, а также внести в них изменения с помощью команды SET.

Если вам требуется задать длинные пути доступа к файлам (командами PATH и APPEND), установить соответствие определенных каталогов некоторым переменным среды, используемым многими пакетами (например, параметр TMP, понимаемый почти любой серьезной программой, начиная от текстовых процессоров и кончая серьезнейшими компиляторами для профессионалов экстра-класса), то вам потребуется увеличить размер области, хранящей информацию о параметрах среды. Для этого нужно добавить в файл CONFIG.SYS строку вроде

```
SHELL = C:\DOS\COMMAND.COM /E:1024 /P
```

что увеличит объем памяти, отведенный под параметры окружения до 1 Кбайта — этого достаточно для подавляющего большинства пользователей. Если и этого объема не хватит, следует увеличить численное значение после ключа /E:. Ключ /P указывает операционной системе на то, что текущий командный процессор является основным и что он не может быть выгружен из памяти командой EXIT.

В MS-DOS 3.20 и более поздних версиях вы можете, при необходимости, увеличить пространство окружения до 32768 байт. Однако следует помнить, что при этом уменьшается доступный объем оперативной памяти.

Существует еще несколько способов, позволяющих справиться с сообщением "Out of environment space". Во-первых, можно объединить каталоги, содержащие выполняемые файлы, после чего путь должен укладываться в одну строку команды PATH. Если это нежелательно или затруднительно, можно попробовать присвоить каталогам более короткие имена, что также позволяет преодолеть неприятности, связанные с нехваткой места для параметров среды.

Кое-что о V20

Кроме базовой версии процессора Intel 8088, существуют различные ее модификации, в частности микропроцессор V20 фирмы NEC. Его часто рекламируют как более быстрый по сравнению с 8088, даже при равной тактовой частоте. Это объясняется использованием более эффективных алгоритмов выполнения команд, экономящих несколько машинных тактов. V20 полностью совместим с 8086 как по системе команд, так и по расположению выводов, что позволяет простой заменой процессора повысить быстродействие вашего компьютера, потратив всего 15 долларов.

В зависимости от алгоритма, используемого при тестировании, повышение производительности составляет от 5 до 30%. Как правило, при решении наиболее типичных задач производительность системы может возрасти в среднем на 10% — не блестящий результат, но вполне достаточный для того, чтобы сделать такое наращивание привлекательным.

О.Липкина
И.Вязаничев

По материалам:
J. Cornell "Battery fixes", PC/Computing, August, 1989



НОВОСТИ

Пользователи настольных типографий, графических пакетов, САПР и других программ, требующих больших объемов дисковой памяти, оценят 5-дюймовые компакт-диски серии Inspire. Они являются безопасными сменными дисками практически неограниченного объема.

Система сочетает в себе возможность замены (как и у обычных гибких дисков), стирания диска, прямого доступа к любой его части, а также большой объем и надежность лазерной технологии.

На каждой стороне диска помещается до 325 Мбайт информации (650 Мбайт на диске).

При работе с большими объемами информации, можно нарастить емкость внешней памяти до 15 Гбайт.

Desktop Magazine, October, 1990

Фирма Hewlett-Packard начала продажу своего самого мощного настольного персонального компьютера. При базовой цене 9145 долларов компьютер Vectra 386/25 PC имеет производительность на 25-50% выше, чем машины типа 80386/20.

Этот компьютер содержит высокоскоростные жесткие диски со встроенным контроллером, графический адаптер HP Super VGA с улучшенной разрешающей способностью, память объемом 2 Мбайта и 32 Кбайта кэш-памяти.

Предлагаются также жесткие диски емкостью 42, 84, 170 и 340 Мбайт со временем доступа 17-19 миллисекунд.

Компьютер предназначен для применения в системах САПР или в качестве сетевого сервера. Для увеличения надежности новая машина использует последние достижения технологии, значительно уменьшающие количество компонентов на плате и, соответственно, потребляемую мощность.

Desktop Magazine, October, 1990

Есть масса программ как для PC, так и для Macintosh, позволяющих сжимать файлы в архивный вид, но до настоящего времени не было программы, которая функционировала бы одинаково в обеих системах. Если раньше можно было разворачивать архивы Stuffit на PC и читать файлы Arc, PKZip и LHarc на Mac, то недавно

появившиеся утилиты LHarc (версия 1.13 для PC и 0.33 — для Mac), разработанные японскими программистами Харюяси Йосикава и Казуаки Ишидзаки, функционируют абсолютно одинаково и в той, и в другой системе.

Изначально утилита появилась на PC. Хотя это и не самая быстрая программа компрессии, она достигает одного из высших уровней сжатия информации, что важно при ее хранении и передаче. Единственное, чего не может делать LHarc — это сегментация больших архивов для размещения их на нескольких дискетах. И что самое важное — программа как для PC, так и для Macintosh бесплатна (то есть для ее использования не требуется дополнительной платы).

Newsbytes News Network, December 24, 1990

Федеральный судья в Огайо Энн Элдрич установила, что действия Джозефа Поппа младшего, местного антрополога, распространявшего компьютерные вирусы по всему миру, являются преступными, и он может быть выдан английским властям для судебного расследования. Приговор направлен госсекретарю, в полномочия которого входит принятие окончательного решения по данному вопросу.

По информации Скотланд-Ярда Попп, отпущеный под залог в 50 тысяч долларов под домашний арест, был задержан сотрудниками ФБР 1 января. Английские сыщики подозревают, что он отправил по почте компаниям, правительственным органам и госпиталям во всем мире до 20000 зараженных вирусом дискет. Предположительно, дискеты получили в Англии, Франции, Западной Германии, Италии, Испании, Дании, Нигерии, ЮАР, Норвегии, Швеции, Швейцарии и Греции.

Дискеты с информацией о вирусе СПИД содержали свой собственный вирус. Этот "троянский конь", оставаясь незамеченным, мог полностью разрушить все данные на компьютере, в который была помещена такая дискета. Для безболезненного удаления вируса с помощью "ключа" получателям рекомендовалось послать чек на 387 долларов в адрес указанного почтового отделения в Панаме.

Адвокат подсудимого заявил, что Попп, который в 1972 году окончил государственный университет в Огайо, а в 1979 году защитил докторскую диссертацию в Гарварде, пытался таким образом собрать денег для исследования вируса СПИД. Он также сообщил, что наклейка на дискете содержала предупреждение о том, что использование дискеты может повредить компьютер. Судья отклонил этот довод, отметив, что даный текст был написан практически нечитаемым мелким шрифтом, и многие пользовались диском, не дочитав наклейку до конца.

Следователь прокуратуры сообщил о своей уверенности в том, что и другие правительства обратятся с аналогичными требованиями о выдаче Поппа. Кроме того, под присягой на судебном заседании было доказано, что информация о СПИДе была весьма поверхностной. Он также заявил, что следствие не располагает материалами, указывающими на благотворительные намерения Поппа.

Newsbytes News Network, December 24, 1990

Новый антивирусный пакет для Macintosh выпустила компания Microcom. Virex version 2.84 обнаруживает 29 вирусов для Мака, в том числе три недавно появившихся — Modm, Zero и новая версия Zuc. Первые два поражают прикладные программы и системные файлы, вызывая частые сбои системы. Новый Zuc поражает только программы, заставляя курсор самопроизвольно двигаться по диагонали экрана. Как только он достигает верхней части экрана, управлять им становится невозможно и единственный способ избавиться от этого — перезагрузка машины.

Компания, по заявлению исполнительного директора, хочет уберечь пользователей от всех известных вирусов для Macintosh и поэтому пристально следит за появлением новой версии этой компьютерной заразы.

Версия 2.84 — это уже девятнадцатое улучшение данной программы с момента ее появления два года назад. Пользователи, подписавшиеся на год (75 фунтов в год), получат новую версию пакета в ближайшем будущем. Тем, кто не сделал этого, новая версия обойдется в 25 фунтов.

Newsbytes News Network, December 24, 1990

Компания Ventura Software начала поставки версии издательского пакета Ventura Publisher для OS/2. Новая версия стоит 895 долларов и работает под управлением OS/2 версии 1.3.

Ventura Publisher существует также в среде Windows 3.0, Macintosh и DOS/GEM. Все версии имеют одинаковый пользовательский интерфейс и могут обмениваться файлами друг с другом.

Лэрри Герхард, президент компании, заявил, что "меньшие требования к памяти в OS/2 версии 1.3 позволили фирме использовать все преимущества в производительности, имеющиеся в операционной системе без снижения характеристик пакета". Он дает возможнотсь вести одновременную обработку нескольких задач. Это позволяет продолжать работу во время печати уже подготовленного файла. Использование виртуальной памяти снимает практически все ограничения на размер обрабатываемых документов.

Newsbytes News Network, December 21, 1990

Советско-американское предприятие «Соваминко»

Рекламно-издательское агентство «КомпьютерПресс»

Принимает заказы на журнал «КомпьютерПресс» и производит отправку наложенным платежом.

Заказ высылается по адресу: 191186, Ленинград, Невский проспект, 28 магазин №1 «Дом книги»

ЗАКАЗ

От кого . . . .

. . . .

Адрес . . . . (почтовый индекс указывать обязательно) . . . .

. . . .

Номера выпусков . . . . **Количество экземпляров** . . . .

На обратной стороне этой страницы помещен бланк заказа на сборник «КомпьютерПресс» Вы можете его вырезать и, заполнив, отправить в конверте по адресу: 113093, Москва, а/я 37.

Подписка на 1991 г. принимается до 1 апреля 1991 г. Число экземпляров — без ограничений.

Вы можете выписать журнал на полгода или на год. Стоимость годовой подписки — 48 рублей, полугодовой — 24 рубля.

Деньги следует перечислить на расчетный счет агентства "КомпьютерПресс".

Банковские реквизиты:
получатель: Автобанк (для зачисления на счет №345708)
расчетный счет получателя: №161202
банк получателя: ЦОУ при Госбанке СССР. МФО №299112.
Копию платежного документа необходимо приложить к бланку заказа.

Без одновременной оплаты подписной стоимости заказ не принимается. Издания агентства "КомпьютерПресс" наложенным платежом не высылаются.

Образец заполнения платежного поручения для предприятий и организаций

	0401002

ПЛАТЕЖНОЕ ПОРУЧЕНИЕ №

199 г.

	ДЕБЕТ	Сумма
Плательщик		
Банк плательщика в г.	сч. №	*48-00*
Получатель *Автобанк (для зачисления на счет 345708)*	КРЕДИТ	
	сч. № 161202	
Банк получателя *ЦОУ при Госбанке СССР в г. Москве МФО № 299112*		
почтой-телеграфом (нужное подчеркнуть)		пеня за из М Р.
Сумма прописью		сумма с пеней
		Вид опер.
Назначение платежа		Назн. плат.
Подписка на сборник “КомпьютерПресс“		Срок плат.
		Очер. плат
		№ гр. банка

М.П.

Подписи клиента

Проведено банком

199 г.

Подписи банка

ЗАКАЗ

От кого ______________________________

Адрес ______________________________

(ПОЧТОВЫЙ ИНДЕКС УКАЗЫВАТЬ ОБЯЗАТЕЛЬНО)

Прошу оформить подписку на 1991 год

Подписная плата в сумме ______________________ перечислена

платежным поручением (почтовым переводом) № ______ от ____________1991 г.

(Копия платежного документа прилагается)



Цена 3 р. 15 к.

ШИРОКИЙ СПЕКТР ПРОГРАММНЫХ ПРОДУКТОВ ВЦ АН СССР — ПОЗВОЛИТ МГНОВЕННО РЕШИТЬ ВАШИ ПРОБЛЕМЫ!

Качество, надежность и сопровождение программных продуктов — наш основной козырь.



СП Диалог

Телефон для справок: (095) 137-01 50
Адрес: 117967, ГСП-1, Москва, ул.Вавилова, д.40, ВЦ АН СССР, комн.103а